目　录

第1章 绪　　论

1.1 滑模变结构控制简介

变结构控制(variable structure control,VSC)本质上是一类特殊的非线性控制,其非线性表现为控制的不连续性。这种控制策略与其他控制的不同之处在于系统的"结构"并不固定,而是可以在动态过程中,根据系统当前的状态(如偏差及其各阶导数等)有目的地不断变化,迫使系统按照预定"滑动模态"的状态轨迹运动,所以又常称变结构控制为滑动模态控制(sliding mode control,SMC),即滑模变结构控制。由于滑动模态可以进行设计且与对象参数及扰动无关,这就使得变结构控制具有快速响应、对参数变化及扰动不灵敏、无需系统在线辨识、物理实现简单等优点。该方法的缺点在于当状态轨迹到达滑模面后,难于严格地沿着滑面向着平衡点滑动,而是在滑模面两侧来回穿越,从而产生颤动。

变结构控制出现于20世纪50年代,经历了50余年的发展,已形成了一个相对独立的研究分支,成为自动控制系统的一种设计方法,适用于线性与非线性系统、连续与离散系统、确定性与不确定性系统、集中参数与分布参数系统、集中控制与分散控制等。并且在实际工程中逐渐得到推广应用,如电机与电力系统控制、机器人控制、飞机控制、卫星姿态控制等。这种控制方法通过控制量的切换使系统状态沿着滑模面滑动,使系统在受到参数摄动和外干扰时具有不变性,正是这种特性使得变结构控制方法受到各国学者的重视。

1.2 变结构控制发展历史

变结构控制的发展过程大致可分为三个阶段。

(1) 1957—1962年

此阶段为研究的初级阶段。前苏联的学者Utkin和Emelyanov在20世纪50年代提出了变结构控制的概念,其基本研究对象为二阶线性系统。

(2) 1962—1970年

60年代的学者开始针对高阶线性系统进行研究,但仍然限于单输入、单输出系统。主要讨论了高阶线性系统在线性切换函数下控制受限与不受限及二次型切换函数的情况。

(3) 1970年以后

在线性空间上研究线性系统的变结构控制。主要结论为变结构控制对摄动及干扰具有不变性。1977年,V. I. Utkin发表了一篇有关变结构控制方面的综述论文[1],提出了滑模变结构控制VSC和滑模控制SMC的方法。此后,各国学者对变结构控制的研究兴趣急剧上升,开始研究多维变结构系统和多维滑动模态,对变结构控制系统的研究由规范空间扩展到更一般的状态空间。K. D. Young等[2]从工程的角度,对滑模控制进行了全面的分析,并

对滑模控制所产生的抖振进行了精确的分析和评估，针对连续系统中的抑制抖振给出了七种解决方法，并针对离散系统在三种情况下的滑模设计进行了分析，为滑模控制在工程上的应用提供了有益的指导。

对变结构控制的研究大多集中在滑动模态上，而对进入切换面之前的运动，即正常的运动段研究较少。中国学者高为炳院士等[3]首先提出了趋近律的概念，列举了诸如等速趋近律、指数趋近律、幂次趋近律直到一般趋近律，高氏等还首次提出了自由递阶的概念。

在解决十分复杂的非线性系统的综合问题时，变结构系统理论作为一种综合方法得到重视。但是滑模变结构对系统的参数摄动和外部干扰的不变性是以控制量的高频抖振换取的，由于在实际应用中，这种高频抖振在理论上是无限快的，没有任何执行机构能够实现；同时，这样的高频输入很容易激发系统的未建模特性，从而影响系统的控制性能。因而抖振现象给变结构控制在实际系统中的应用带来了困难。

由于人们认识到变结构系统中的滑动模态具有不变性，这种理想的鲁棒性对工程应用也是很有吸引力的。高精度伺服系统存在着许多不利于控制系统设计的因素，如非线性因素、外干扰及参数摄动等。由于离散滑模变结构控制自身的缺点，将其直接应用到高精度的伺服系统中将会有一定的困难，因为控制输出的高频抖振会损坏伺服系统中的电机和其他设备。要将离散滑模变结构控制应用到伺服系统中，使其真正发挥它的强鲁棒性，必须对传统的离散滑模变结构控制进行改进，并针对抖振现象改进离散滑模控制器，将有害的抖振减小到一定程度，并且又要保证滑模控制的不变性。因此，对传统的离散滑模变结构控制的改进、抖振的削弱成为研究的重点。

1.3 滑模变结构控制基本原理

滑模变结构控制是变结构控制系统的一种控制策略。这种控制策略与常规控制的根本区别在于控制的不连续性，即一种使系统“结构”随时间变化的开关特性。该控制特性可以迫使系统在一定特性下沿规定的状态轨迹作小幅度、高频率的上下运动，即所谓的“滑动模态”或“滑模”运动。这种滑动模态是可以设计的，且与系统的参数及扰动无关。这样，处于滑模运动的系统就具有很好的鲁棒性。

滑动模态控制的概念和特性如下：

(1) 滑动模态定义及数学表达

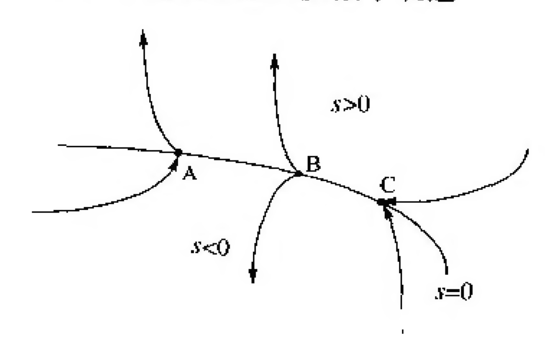


图 1-1 切换面上三种点的特性

考虑一般的情况，在系统

$$\dot{x} = f(x) \qquad x \in \mathbf{R}^n \tag{1.1}$$

的状态空间中，有一个切换面 $s(x) = s(x_1, x_2, \cdots, x_n) = 0$，它将状态空间分成上下两部分 $s>0$ 及 $s<0$。在切换面上的运动点有三种情况，如图 1-1 所示。

通常点—— 系统运动点运动到切换面 $s=0$ 附近时，穿越此点而过(点 A)；

起始点——系统运动点到达切换面 $s=0$ 附

近时，从切换面的两边离开该点（点 B）；

终止点——系统运动点到达切换面 $s=0$ 附近时，从切换面的两边趋向于该点（点 C）。

在滑模变结构中，通常点与起始点无多大意义，而终止点却有特殊的含义。因为如果在切换面上某一区域内所有的运动点都是终止点，则一旦运动点趋近于该区域，就会被"吸引"到该区域内运动。此时，称在切换面 $s=0$ 上所有的运动点都是终止点的区域为"滑动模态"区，或简称为"滑模"区。系统在滑模区中的运动就叫做"滑模运动"。

按照滑动模态区上的运动点都必须是终止点这一要求，当运动点到达切换面 $s(x)=0$ 附近时，必有

$$\lim_{s\to 0^+}\dot{s}\leqslant 0 \quad 及 \quad \lim_{s\to 0^-}\dot{s}\geqslant 0 \tag{1.2}$$

或者

$$\lim_{s\to 0^+}\dot{s}\leqslant 0\leqslant \lim_{s\to 0^-}\dot{s} \tag{1.3}$$

式(1.3)也可写成

$$\lim_{s\to 0}s\dot{s}\leqslant 0 \tag{1.4}$$

此不等式对系统提出了一个形如

$$v(x_1,x_2,\cdots,x_n)=[s(x_1,x_2,\cdots,x_n)]^2 \tag{1.5}$$

的李雅普诺夫（Lyapunov）函数的必要条件。由于在切换面邻域内函数式(1.5)是正定的，而按照式(1.4)，s^2 的导数是负半定的，也就是说在 $s=0$ 附近 v 是一个非增函数，因此，如果满足条件式(1.4)，则式(1.5)是系统的一个条件李雅普诺夫函数。系统本身也就稳定于条件 $s=0$。

(2) 滑模变结构控制的定义

滑模变结构控制的基本问题如下：

设有一控制系统

$$\dot{x}=f(x,u,t) \qquad x\in \mathbf{R}^n, u\in \mathbf{R}^m, t\in \mathbf{R} \tag{1.6}$$

需要确定切换函数

$$s(x), \qquad s\in \mathbf{R}^m \tag{1.7}$$

求解控制函数

$$u=\begin{cases}u^+(x) & s(x)>0\\ u^-(x) & s(x)<0\end{cases} \tag{1.8}$$

其中，$u^+(x)\neq u^-(x)$，使得

① 滑动模态存在，即式(1.8)成立；

② 满足可达性条件，在切换面 $s(x)=0$ 以外的运动点都将于有限的时间内到达切换面；

③ 保证滑模运动的稳定性；

④ 达到控制系统的动态品质要求。

上面的前三点是滑模变结构控制的三个基本问题，只有满足了这三个条件的控制才叫滑模变结构控制。

1.4　滑模变结构控制理论的研究方向

1.4.1　滑模变结构控制系统的抖振问题

从理论角度上讲，在一定意义上，由于滑动模态可以按需要设计，而且系统的滑模运动与控制对象的参数变化和系统的外界干扰无关，因此滑模变结构控制系统的鲁棒性要比一般常规的连续系统强。然而，滑模变结构控制在本质上的不连续开关特性将会引起系统的抖振。

对于一个理想的滑模变结构控制系统，假设"结构"切换的过程具有理想开关特性（即无时间和空间滞后），系统状态测量精确无误，控制量不受限制，则滑动模态总是降维的光滑运动而且渐近稳定于原点，不会出现抖振。但是对于一个现实的滑模变结构控制系统，这些假设是不可能完全成立的。特别是对于离散系统的滑模变结构控制系统，都将会在光滑的滑模面上叠加一个锯齿形的轨迹。于是，在实际系统中，抖振是必定存在的，而且若消除了抖振，也就消除了变结构控制的抗摄动和抗扰动的能力，因此，消除抖振是不可能的，只能在一定程度上削弱它。抖振问题成为变结构控制在实际系统中应用的突出障碍。

抖振产生的主要原因有：

(1) 时间滞后开关

在切换面附近，由于开关的时间滞后，控制作用对状态的准确变化被延迟一定的时间；又因为控制量的幅度是随着状态量的幅度逐渐减少的，所以表现为在光滑的滑模面上叠加一个衰减的三角波。

(2) 空间滞后开关

开关的空间滞后相当于在状态空间中存在一个状态量变化的"死区"。因此，其结果是在光滑的滑模面上叠加了一个等幅波形。

(3) 系统惯性的影响

由于任何物理系统的能量不可能无限大，因而系统的控制力也不能无限大，这就使系统的加速度有限；另外，系统惯性总是存在的，所以使得控制切换伴有滞后，这种滞后与时间滞后效果相同。

(4) 离散系统本身造成的抖振

离散系统的滑动模态是一种"准滑动模态"，它的切换动作不是正好发生在切换面上，而是发生在以原点为顶点的一个锥形体的表面上。因此有衰减的抖振，而且锥形体越大，抖振幅度越大。该锥形体的大小与采样周期有关。

总之，抖振产生的原因在于：当系统的轨迹到达切换面时，其速度是有限大，惯性使运动点穿越切换面，从而最终形成抖振，叠加在理想的滑动模态上。对于实际的计算机采样系统而言，计算机的高速逻辑转换以及高精度的数值运算使得切换开关本身的时间及空间滞后影响几乎不存在，因此，开关的切换动作所造成的控制的不连续性是抖振发生的根本原因。

在实际系统中，由于时间滞后开关、空间滞后开关、系统惯性、系统延迟及测量误差等因

素，使变结构控制在滑动模态下伴随着高频抖振，抖振不仅影响控制的精确性，增加能量消耗，而且系统中的高频未建模动态很容易被激发起来，破坏系统的性能，甚至使系统产生振荡或失稳，损坏控制器部件。因此，关于变结构控制信号抖振消除的研究成为变结构控制研究的首要问题。

国内外针对滑模控制抗抖振问题的研究很多，许多学者都从不同的角度提出了解决方法。目前，有代表性的研究工作主要有以下几方面。

(1) 准滑动模态方法

20世纪80年代Slotine等[4]在滑动模态控制的设计中引入了"准滑动模态"和"边界层"的概念，实现准滑动模态控制，采用饱和函数代替切换函数，即在边界层外采用正常的滑模控制，在边界层内为连续状态的反馈控制，有效地避免或削弱了抖振，为变结构控制的工程应用开辟了道路。此后，有许多学者对切换函数和边界层的设计进行了研究。

① 连续函数近似法

S. C. Y Chung等[5]采用Sigmoid连续函数来代替切换函数。J. X. Xu等[6]针对直流电机伺服系统的未建模动态进行了分析和描述，设计了基于插补平滑算法的滑模控制器，实现了非连续切换控制的连续化，有效地消除了未建模动态对直流电机伺服系统造成的抖振。

② 边界层的设计

边界层厚度越小，控制效果越好，但同时又会使控制增益变大，抖振增强；反之，边界层厚度越大，抖振越小，但又会使控制增益变小，控制效果变差。为了获得最佳抗抖振效果，边界层厚度应自适应调整。

K. Erbatur等[7]提出了一种高增益滑模控制器，设控制信号输入为u，切换函数为s，将$|\dot{u}|$作为衡量抖振的指标，按降低控制抖振来设计模糊规则，将$|s|$和$|\dot{u}|$作为模糊规则的输入，模糊推理的输出为边界层厚度的变化，实现了边界层厚度的模糊自适应调整。M. S. Chen等[8]针对不确定性线性系统，同时考虑了控制信号的降抖振与跟踪精度的要求，提出了一种基于系统状态范数的边界层厚度在线调整算法。P. V. Vicente等[9]提出了一种新型的动态滑模控制，采用饱和函数方法，通过设计一种新型非线性切换函数s，消除了滑模到达阶段的抖振，实现了全局鲁棒滑模控制，有效地解决了一类非线性机械系统的控制抖振问题。S. Seshagiri等[10]为了减小边界层厚度，在边界层内采用了积分控制，既获得了稳态误差，又避免了抖振。边界层的方法仅能保证系统状态收敛到以滑动面为中心的边界层内，只能通过较窄的边界层来任意地接近滑模，但不能使状态收敛到滑模。

(2) 趋近律方法

高为炳[3]利用趋近律的概念，提出了一种变结构控制系统的抖振消除方法。以指数趋近律$\dot{s}=-\varepsilon\mathrm{sgn}(s)-ks$为例，通过调整趋近律的参数$k$和$\varepsilon$，既可以保证滑动模态到达过程的动态品质，又可以减弱控制信号的高频抖振，但较大的ε值会导致抖振。翟长连等[11]分析了指数趋近律应用于离散系统时趋近系数造成抖振的原因，并对趋近系数与抖振的关系进行了定量的分析，提出了趋近系数ε的自适应调整算法。于双和等[12]提出了将离散趋近律与等效控制相结合的控制策略，离散趋近律仅在趋近阶段起作用，当系统状态到达准滑模模态阶段，采用抗干扰的离散等效控制，这样既保证了趋近模态具有良好的品质，又降低了准滑动模态带，消除了抖振。K. Jiang等[13]将模糊控制应用于指数趋近律中，通过分析切换函数与指数趋近律中系数的模糊关系，利用模糊规则调节指数趋近律的系数，其中切换函数

的绝对值$|s|$作为模糊规则的输入，指数趋近律的系数ε和k作为模糊规则的输出，使滑动模态的品质得到了进一步的改善，消除了系统的高频抖振。

(3) 滤波方法

通过采用滤波器，对控制信号进行平滑滤波，是消除抖振的有效方法。

W. C. Su等[14]为了消除离散滑模控制的抖振，设计了两种滤波器：前滤波器和后滤波器，其中前滤波器用于控制信号的平滑及缩小饱和函数的边界层厚度，后滤波器用于消除对象输出的噪声干扰。P. Kachroo等[15]在边界层内，对切换函数$s(x)$采用了低通滤波器，得到平滑的$s(x)$信号，并采用了内模原理，设计了一种新型的带有积分和变边界层厚度的饱和函数，有效地降低了抖振。B. P. Kang等[16]利用机器人的物理特性，通过在控制器输出端加入低通滤波器，设计了虚拟滑模控制器，实现了机器人全鲁棒变结构控制，并保证了系统的稳定，有效地消除了抖振。H. Yanada等[17]设计了带有滤波器的变结构控制器，有效地消除了控制信号的抖振，得到了抑制高频噪声的非线性控制器，实现了存在未建模动态的电液伺服马达的定位控制。D. Krupp等[18]为了克服未建模动态特性造成的滑动模态抖振，设计了一种新型滑模控制器，该控制器输出通过一个二阶滤波器，实现控制器输出信号的平滑，其中辅助滑动模面s的系数通过滑模观测器得到。J. X. Xu等[19]提出了一种新型控制律，即$u=u_e+K(t)u_s+u_c$，该控制律由三部分构成，即等效控制、切换控制和连续控制。在控制律中采用了两个低通滤波器，其中通过一个低通滤波器得到切换项的增益$K(t)$，通过另一个低通滤波器得到等效控制项$u_e(t)$，并进行了收敛性和稳定性分析，有效地抑制了抖振，实现了多关节机器手的高性能控制。

(4) 观测器方法

在常规滑模控制中，往往需要很大的切换增益来消除外加干扰及不确定项，因此，外界干扰及不确定项是滑模控制中抖振的主要来源。利用观测器来消除外界干扰及不确定项成为解决抖振问题研究的重点。

A. Kawamura等[20]为了将常规滑模控制方法应用于带有较强外加干扰的伺服系统中，设计了一种新型干扰观测器，通过对外加干扰的前馈补偿，大大降低了滑模控制器中切换项的增益，有效地消除了抖振。Y. S. Kim等[21]在滑模控制中设计了一种基于二元控制理论的干扰观测器，将观测到的干扰进行前馈补偿，减小了抖振。H. Liu[22]提出了一种基于误差预测的滑模控制方法，在该方法中设计了一种观测器和滤波器，通过观测器消除了未建模动态的影响，采用均值滤波器实现了控制输入信号的平滑，有效地消除了未建模动态造成的抖振。Y. Eun等[23]设计了一种离散的滑模观测器，实现了对控制输入端干扰的观测，从而实现对干扰的有效补偿，相对减小了切换增益。宋立忠等[24]提出了一种新型离散趋近律，其特点是可以使系统状态稳定于原点，并针对系统的不确定部分设计了扰动预测器，对常值或变化率较慢的扰动具有很高的估计精度，有效地减弱了抖振。

(5) 动态滑模方法

传统的滑模控制方法中切换函数一般只依赖于系统状态，与控制输入无关，不连续项会直接转移到控制器中。动态滑模方法将常规变结构控制中的切换函数s通过微分环节构成新的切换函数σ，该切换函数与系统控制输入的一阶或高阶导数有关，可将不连续项转移到控制的一阶或高阶导数中去，得到在时间上本质连续的动态滑模控制律，有效地降低了抖振。

G. Bartolini 等[25~28]通过设计切换函数的二阶导数，实现了对带有未建模动态和不确定性机械系统的无抖振滑模控制，并将该方法扩展到多输入系统中。通过采用动态滑模控制器，得到在时间上本质连续的动态变结构控制律，有效地消除了抖振，该方法已成功地应用于带有库仑摩擦的机械系统及机器人力臂控制系统中。M. Hamerlain 等[29]将动态滑模控制用于机器人力臂的控制，有效地消除了抖振。晁红敏等[30]采用动态滑模控制实现了移动机器人的跟踪控制，明显地消除了抖振。

(6) 模糊方法

根据经验，以降低抖振来设计模糊规则，可有效地降低滑模控制的抖振。模糊滑模控制柔化了控制信号，即将不连续的控制信号连续化，可减轻或避免一般滑模控制的抖振现象。模糊逻辑还可以实现滑模控制参数的自调整。

Q. P. Ha 等[31]采用等效控制、切换控制和模糊控制三部分构成模糊滑模控制器，在模糊控制器中，通过模糊规则的设计，降低了切换控制的影响，有效地降低了抖振。K. Y. Zhuang 等[32]利用模糊控制对系统的不确定项进行在线估计，实现切换增益的模糊自调整，在保证滑模到达条件满足的情况下，尽量减小切换增益，以降低抖振。S. H. Ryu 等[33]建立了滑模控制的抖振指标，以降低抖振来设计模糊规则。模糊规则的输入为当前的抖振指标值，模糊规则的输出为边界层厚度变化，通过模糊推理，实现了边界层厚度的自适应调整。张天平等[34]提出了一种基于模糊逻辑的连续滑模控制方法，使用连续的模糊逻辑切换代替滑模控制的非连续切换，避免了抖振。孙宜标等[35]提出了一种基于模糊自学习的滑模变结构控制方法，控制器输出为 $u=u_{eq}+u_{vss}$，即采用模糊滑模控制器来代替滑模控制的切换部分 u_{vss}，保证了控制律的连续性，通过模糊基函数的自学习，达到满足滑模存在条件和减少抖振的目的。

(7) 神经网络方法

H. Morioka 等[36]采用神经网络实现了对线性系统的非线性部分、不确定部分和未知外加干扰的在线估计，实现了基于神经网络的等效控制，有效地消除了抖振。M. Ertugrul 等[37]提出了一种新型神经网络滑模控制方法，采用两个神经网络分别逼近等效滑模控制部分及切换滑模控制部分，无需对象的模型，有效地消除了控制器的抖振，该方法已成功地应用于机器人的轨迹跟踪。S. J. Huang 等[38]利用神经网络的逼近能力，设计了一种基于 RBF 神经网络的滑模控制器，将切换函数 s 作为网络的输入，控制器完全由连续的 RBF 函数实现，取消了切换项，消除了抖振。达飞鹏等[39]将滑模控制器分为两部分，一部分为神经网络滑模控制器，另一部分为线性反馈控制器，利用模糊神经网络的输出代替滑模控制中的符号函数，保证了控制律的连续性，从根本上消除了抖振。

(8) 遗传算法优化方法

遗传算法是建立在自然选择和自然遗传学机理基础上的迭代自适应概率性搜索算法，在解决非线性问题时表现出很好的鲁棒性、全局最优性、可并行性和高效率，具有很高的优化性能。

K. C. Ng 等[40]针对非线性系统设计了一种软切换模糊滑模控制器，采用遗传算法对该控制器增益参数及模糊规则进行离线优化，有效地减小了控制增益，从而消除了抖振。

F. J. Lin 等[41]针对不确定性伺服系统设计了一种积分自适应滑模控制器，通过该控制器中的自适应增益项来消除不确定性及外加干扰。如果增益项为常数，则会造成抖振。为此，F. J. Lin 等开发了一种实时遗传算法，实现了滑模变结构控制器中自适应增益项的在线自适应优化，有效地减小了抖振。C. F. Zhang 等[42]采用遗传算法进行切换函数的优化，将抖振的大小作为优化适应度函数的重要指标，构造一个抖振最小的切换函数。

(9) 切换增益方法

由于抖振主要是由控制器的不连续切换项造成，因此，减小切换项的增益，便可有效地消除抖振。

C. L. Hwang[43]根据滑模控制的 Lyapunov 稳定性要求，设计了时变的切换增益，减小了抖振。L. J. Wong 等[44]对切换项进行了变换，通过设计一个自适应积分项来代替切换项，实现了切换项增益的自适应调整，有效地减小了切换项的增益。林岩等[45]针对一类带有未建模动态系统的控制问题，提出了一种鲁棒低增益变结构模型参考自适应控制新方法，使系统在含未建模动态时所有辅助误差均可在有限时间内收敛为零，并保证在所有情况下均为低增益控制。F. J. Lin 等[46]提出了采用模糊神经网络的切换增益自适应调节算法，当跟踪误差接近于零时，切换增益接近于零，大大降低了抖振。

(10) 扇形区域法

J. X. Xu 等[47]针对不确定非线性系统，设计了包含两个滑动模面的滑动扇区，构造连续切换控制器，使得在开关面上控制信号是连续的。D. Y. Yang 等[48]采用滑动扇区法，在扇区之内采用连续的等效控制，在扇区之外采用趋近律控制，很大程度地消除了控制的抖振。

(11) 其他方法

Y. Konno 等[49]针对滑模变结构控制中引起抖振的动态特性，将抖振看成叠加在理想滑模上的有限频率的振荡，提出了滑动切换面的 H_∞ 优化设计方法，即通过切换面的设计，使滑动模态的频率响应具有某种希望的形状，实现频率整形。该频率整形能够抑制滑动模态中引起抖振的频率分量，使切换面为具有某种"滤波器"特性的动态切换面。C. Edwards[50]设计了一种能量函数，该能量函数包括控制精度和控制信号的大小，采用 LMI (linear matrix inequality，线性矩阵不等式)的方法设计滑动模面，使能量函数达到最小，实现了滑动模面的优化，提高了控制精度，消除了抖振。

上述各种方法中，每种方法都有各自的优点和局限性，针对具体的问题需要进行具体的分析。

(1) 针对不同的问题，需要采用不同的方法。例如，趋近律方法在不确定性及干扰小的情况下会有很好的降抖振效果，在不确定性或干扰较大时，需要采用其他方法。

(2) 对于同一问题，可以采用不同的方法。例如，对于外加干扰引起的抖振，可以采用动态滑模方法来消除抖振，或采用变切换增益法来降低抖振。

(3) 每种方法都有各自的局限性。针对复杂的控制问题，需要各种方法相互结合、相互补充，才能达到理想的无抖振滑模控制。例如采用模糊或神经网络方法可实现摩擦补偿，采用干扰观测器法可消除干扰造成的抖振，采用滤波法可消除未建模动态造成的抖振，采用准滑动模态法可进一步降低抖振。又如，利用遗传算法来优化模糊规则或神经网络，可达到消除抖振的最佳效果。

1.4.2 离散系统滑模变结构控制

连续时间系统和离散时间系统的控制有很大差别。自20世纪80年代初至今，由于计算机技术的飞速发展，实际控制中使用的都是离散系统，因此，对离散系统的变结构控制研究尤为重要。对离散系统变结构控制的研究是从20世纪80年代末开始的，例如，S. Z. Sarpturk等[51]于1987年提出了一种新型离散滑模到达条件，并在此基础上提出了离散控制信号必须是有界的理论；K. Furuta[52]于1990年提出了基于等效控制的离散滑模变结构控制；高为炳[53]于1995年提出了基于趋近律的离散滑模变结构控制。他们各自提出的离散滑模变结构滑模存在条件及其控制方法已被广泛应用。

然而，传统设计方法存在两方面不足：一是由于趋近律自身参数及切换开关的影响，即使对名义系统，系统状态轨迹也只能稳定于原点邻域的某个抖振；二是由于根据不确定性上下界进行控制器设计，可能会造成大的反馈增益，使控制抖振加剧。近年来国内外学者对离散系统滑模变结构控制的研究不断深入。Y. D. Pan等[54]提出了基于**PR**型的离散系统滑模面设计方法，其中**P**和**R**分别为与系统状态有关的正定对称阵和半正定对称阵，在此基础上设计了稳定的离散滑模控制器，通过适当地设计**P**和**R**，保证了控制器具有良好的性能。A. J. Koshkouei等[55]针对离散系统提出了一种新型滑模存在条件，进一步拓展了离散滑模控制的设计，在此基础上设计了一种新型滑模控制律。针对离散系统中滑模控制的不变性和鲁棒性难以有效保证，J. H. Kim等[56]提出了三种解决方法，在第一种方法中，采用了干扰补偿器和解耦器消除干扰；在第二种方法中，采用回归切换函数方法来消除干扰；在第三种方法中，采用回归切换函数和解耦器相结合的方法来消除干扰。上述三种方法已成功地应用于数控中。S. V. Emelyanov等[57]针对数字滑模控制的鲁棒性进行了系统的研究，设计出了高增益数字滑模控制器。

1.4.3 自适应滑模变结构控制

自适应滑模变结构控制是滑模变结构控制与自适应控制的有机结合，是解决参数不确定或时变参数系统控制问题的一种新型控制策略。R. H. Sira等[58]针对线性化系统将自适应Backstepping与滑模变结构控制设计方法结合在一起，实现了自适应滑模变结构控制。M. R. Bolivar等[59]针对一类最小相位的可线性化的非线性系统，设计了一种动态自适应变结构控制器，实现了带有不确定性和未知外干扰的非线性系统鲁棒控制。在一般的滑模变结构控制中，为了保证系统能够达到切换面，在设计控制律时通常要求系统不确定性范围已知，这个要求在实际工程中往往很难达到。针对具有未知参数变化和干扰变化的不确定性系统的变结构控制，F. J. Lin等[60]设计了一种新型的带有积分的滑动模面，并采用一种自适应滑模控制方法，控制器的设计无需不确定性及外加干扰的上下界，实现了一类不确定伺服系统的自适应变结构控制。针对自适应滑模控制中参数估计值无限增大的缺点，G. Wheeler等[61]提出了一种新的参数自适应估计方法，保证了变结构控制增益的合理性。

近年来，变结构模型参考自适应控制（VSS-MRAC）理论取得了一系列重要进展，由于该方法具有良好的过渡过程性能和鲁棒性，在工程上得到了很好的应用。N. Bekiroglu

等[62]设计了一种新型动态滑动模面，滑动模面参数通过采用自适应算法估计得到，从而实现了非线性系统的模型参考自适应滑模控制。J. B. Song 等[63]针对一类不确定性气压式伺服系统，提出了模型参考自适应滑模控制方法，并在此基础上提出了克服抖振的有效方法。

1.4.4 非匹配不确定性系统的滑模变结构控制

由于大多数系统不满足变结构控制的匹配条件，因此，存在非匹配不确定性系统的变结构控制成为一个研究重点。C. M. Kwan[64]利用参数自适应控制方法，构造了一个变参数的切换函数，对具有非匹配不确定性的系统进行了变结构控制设计。采用基于线性矩阵不等式 LMI 的方法，为非匹配不确定性系统的变结构控制提供了新的思路，H. H. Choi[65~67]针对非匹配不确定性系统，专门研究了利用 LMI 方法进行变结构控制设计的问题。Backstepping 方法通过引入中间控制器，使控制器的设计系统化、程序化，它对于非匹配不确定性系统及非最小相位系统的变结构控制是一种十分有效的方法。采用 Backstepping 设计方法，J. Li[68]实现了对于一类具有非匹配不确定性的非线性系统的变结构控制。将 Backstepping 设计方法、滑模控制及自适应方法相结合，A. J. Koshkouei 等[69]实现了一类具有非匹配不确定性的非线性系统的自适应滑模控制。

1.4.5 针对时滞系统的滑模变结构控制

由于实际系统普遍存在状态时滞、控制变量时滞问题，因此，研究具有状态或控制时滞系统的变结构控制，对进一步促进变结构控制理论的应用具有重要意义。F. Gouaisbaut 等[70]对于具有输入时滞的不确定性系统，通过状态变换的方法，实现了滑模变结构控制器的设计。C. H. Chou 等[71]研究了带有时滞关联项的大系统的分散模型跟踪变结构控制问题，其中被控对象的时滞关联项必须满足通常的匹配条件。Y. Q. Xia 等[72]采用趋近律的方法设计了一种新型控制器，采用基于 LMI 的方法进行了稳定性分析和切换函数的设计，所设计的控制器保证了对非匹配不确定性和匹配的外加干扰具有较强的鲁棒性，解决了非匹配不确定性时滞系统的变结构控制问题。Y. B. Shtessel 等[73]针对带有输出延迟非线性系统的滑模控制器的设计进行了探讨，在该方法中，将延迟用一阶 Pade 近似的方法来代替，并将非最小相位系统转化为稳定系统，在存在未建模动态和延迟不确定性条件下，控制器获得了很好的鲁棒性能。

1.4.6 非线性系统的滑模变结构控制

非线性系统的滑模变结构控制一直是人们关注的热点。V. I. Utkin[74]研究了具有正则形式的非线性系统的变结构控制问题，为非线性系统变结构控制理论的发展奠定了基础。目前，非最小相位非线性系统、输入受约束非线性系统、输入和状态受约束非线性系统等复杂问题的变结构控制是该领域研究的热点。Y. F. Chang 等[75]将 Anti-windup 方法与滑模控制方法相结合，设计了输入饱和的 Anti-windup 算法，可以实现当输出为饱和时的高精度变结构控制，X. Y. Lu 等[76]利用滑模变结构控制方法实现了一类非最小相位非线性系统的

鲁棒控制，J. Wang 等[77]利用输入输出反馈线性化、相对度、匹配条件等非线性系统的概念，采用输出反馈变结构控制方法实现了一类受约束非线性系统的鲁棒输出跟踪反馈控制。G. Bartolini 等[78]利用 Backstepping 方法，实现了非线性不确定性系统的变结构控制。

1.4.7 Terminal 滑模变结构控制

在普通的滑模控制中，通常选择一个线性的滑动超平面，使系统到达滑动模态后，跟踪误差渐进地收敛为零，并且渐进收敛的速度可以通过选择滑模面参数矩阵任意调节。尽管如此，无论如何状态跟踪误差都不会在有限时间内收敛为零。

近年来，为了获得更好的性能，一些学者提出了一种 Terminal 滑模控制策略[79~81]，该策略在滑动超平面的设计中引入了非线性函数，使得在滑模面上跟踪误差能够在有限时间内收敛到零。Terminal 滑模控制是通过设计一种动态非线性滑模面方程实现的，即在保证滑模控制稳定性的基础上，使系统状态在指定的有限时间内达到对期望状态的完全跟踪。例如，文献[82]将动态非线性滑模面方程设计为 $s=x_2+\beta x_1^{q/p}$，其中 $p>q$，q 和 p 均为正的奇数，$\beta>0$。但该控制方法由于非线性函数的引入使得控制器在实际工程中实现困难，而且如果参数选取不当，还会出现奇异问题。Y. Feng 等[83]探讨了非奇异 Terminal 滑模控制器的设计问题，并针对 N 自由度刚性机器人的控制进行了验证。C. W. Tao 等[84]采用模糊规则设计了 Terminal 滑模控制器的切换项，并通过自适应算法对切换项增益进行自适应模糊调节，实现了非匹配不确定性时变系统的 Terminal 滑模控制，同时降低了抖振。文献[85]中只对一个二阶系统给出了相应的 Terminal 滑面，滑模面的导数是不连续的，不适用于高阶系统。庄开宇等[86]设计了一种适用于高阶非线性系统的 Terminal 滑面，克服了文献[85]中的滑模面导数不连续的缺点，并消除了滑模控制的到达阶段，确保了系统的全局鲁棒性和稳定性。进一步地，庄开宇等[87]又针对系统参数摄动和外界扰动等不确定性因素上界的未知性，实现了 MIMO 系统的自适应 Terminal 控制器设计，所设计的滑模面方程为 $\sigma(\boldsymbol{X},t)=\boldsymbol{CE}-\boldsymbol{W}(t)$，其中 $\boldsymbol{W}(t)=\boldsymbol{CP}(t)$，误差向量为 $\boldsymbol{E}$，$\boldsymbol{C}=[c_1\quad c_2\quad \cdots\quad c_n]$为常数矩阵，$\boldsymbol{P}(t)$为与误差及时间有关的函数矩阵。

1.4.8 全鲁棒滑模变结构控制

在变结构控制系统中，系统的运动可分为两个阶段：第一阶段是到达运动阶段，即滑模控制中的趋近过程，在该过程中，由到达条件保证系统运动在有限时间内从任意初始状态到达切换面；第二阶段是系统在控制律的作用下保持滑模运动。由于变结构控制的优点在于其滑动模态具有鲁棒性，即系统只有在滑动阶段才具有对参数摄动和外界干扰的不敏感性。如果能缩短到达滑模时间，将有效地改善系统的动态性能，而如何缩短到达时间则是变结构控制的一个重要研究方向。

全滑模控制器为具有全程滑动模态的变结构控制器，在该控制器的作用下，消除滑模控制的到达运动阶段，使系统在响应的全过程都具有鲁棒性，克服了传统变结构控制中到达模态不具有鲁棒性的特点。全局滑模控制是通过设计一种动态非线性滑模面方程来实现的，即在保证滑模控制稳定性的基础上，消除滑模控制中的趋近过程。动态非线性滑模面方程

设计为 $s=\dot{e}+ce-f(t)$，其中 e 为跟踪误差，c 为正的常数，函数 $f(t)$ 满足以下三个条件：

(1) $f(0)=\dot{e}_0+ce_0$；

(2) 当 $t\to\infty$ 时，$f(t)\to 0$；

(3) $\dot{f}(t)$ 存在且有界。

通过上述设计，使控制器在稳定条件下具有全局鲁棒性(global)。

Y. S. Lu 等[88]提出了一种全局鲁棒的滑模控制器 GSMC，在该控制器中，考虑了对象的不确定性、外加干扰。针对控制输入信号的限制，对滑模线进行了优化设计，使对象按理想的轨迹跟踪，并有效地利用了电机的输出。该控制器成功地应用于直流无刷电机的控制中。H. S. Choi 等[89]在 Y. S. Lu 研究的基础上对滑模线进行了改进，使对象在有限的控制输入内沿着理想的轨迹运行，并按最短时间到达。肖雁鸿等[90]基于滑模运动方程与系统期望特性的等价性设计了一种非线性切换函数，提出了一类具有全程滑动模态的变结构控制器，即全滑模控制器，使系统在响应的全过程都具有鲁棒性，克服了传统变结构控制中到达模态不具有鲁棒性的特点。米阳等[91]针对一类具有不确定性离散系统，设计出了全鲁棒滑模控制器，通过选择切换函数 $s(x)$，使系统轨线一开始就落在切换面上。

1.4.9 滑模观测器的研究

通过滑模观测器，可实现状态部分可测或完全不可测情况下的控制。利用滑模变结构方法设计非线性观测器是一个重要的研究方向[92]。Y. H. Zheng 等[93]设计了一种非线性自适应滑模观测器，并将该观测器用于电机控制中。C. P. Tan 等[94]将滑模观测器用于解决对传感器的故障诊断和重构问题。S. M. Kim 等[95]采用自适应滑模观测器，实现了电机定子电流和转子流量的精确估计，在滑模观测器中采用了 H_∞ 方法，使观测精度得到了提高。S. S. Lee 等[96]采用滑模观测器实现了状态方程中未知参数的估计，在滑模观测器中采用了 H_∞ 方法，降低了滑模控制器的增益。

1.4.10 神经滑模变结构控制

神经网络是一种具有高度非线性的连续时间动力系统，它有着很强的自学习功能和对非线性系统的强大映射能力。神经网络用于滑模变结构控制，可实现自适应滑模控制。

S. C. Lin 等[97]将传统方法与神经网络相结合，无需对象的精确模型，设计了基于 RBF (radial basis function，径向基函数)网络的滑模控制器，该控制器成功地应用于非线性单级倒立摆的自适应控制。D. Munoz 等[98]针对一类非线性离散机器人力臂系统设计了自适应滑模控制器，在该控制器中采用两个神经网络实现了非线性系统中未知函数部分的逼近，从而实现了基于神经网络的滑模自适应控制。Z. H. Man 等[99]提出了一种新型神经网络滑模控制方法，采用 RBF 网络辨识系统不确定部分的上界，该方法已成功地应用于机器手的轨迹跟踪。J. M. Carrasco 等[100]采用 BP 网络代替带有切换的滑模控制器，通过神经网络权值的在线调整，实现了针对变频器的抗抖振自适应滑模控制。G. Parma 等[101]将 BP 网络学习算法与滑模控制相结合，构成新的闭环控制系统，利用 BP 网络的在线学习功能，设计出了一种新型滑模——神经网络控制器，实现了感应电机的自适应滑模控制。

1.4.11 模糊滑模变结构控制

在常规的模糊滑模控制中，控制目标从跟踪误差转化为滑模函数，模糊控制器的输入不是$(e,\dot{e})$而是$(s,\dot{s})$，通过设计模糊规则，使滑模面s为零。模糊滑模控制柔化了控制信号，可减轻或避免一般滑模控制的抖振现象。S. W. Kim 等[102]提出了一种基于模糊滑面的模糊控制，将滑模面进行模糊划分，设计了基于稳定的模糊控制器。B. Yoo 等[103]利用模糊系统逼近未知函数，只要知道未知函数的边界，便可设计基于模糊的自适应滑模控制器。Y. S. Lu 等[104]采用模糊系统的输出代替滑模等效控制，并通过模糊自适应学习，使模糊系统的输出渐进逼近滑模等效控制。C. Y. Liang 等[105]将控制器设计为$u=\hat{u}+K_f(s,\dot{s})$的形式，滑模面s采用一种积分滑模函数的形式，K_f为基于$(s,\dot{s})$输入的滑模模糊控制系统的输出。R. J. Wai等[106]在滑动模面中加入了积分项，通过模糊规则设计了模糊滑模控制器，并通过自适应算法实现了对切换项系数的自适应估计。J. Y. Chen[107]采用模糊规则设计了基于等效控制的模糊滑模控制器，其中控制器的切换项增益通过隶属函数来调节；为了降低抖振，设计了抖振指标，通过采用遗传算法来优化隶属函数，实现了抖振的消除。

1.4.12 积分滑模变结构控制

普通的滑模变结构控制在跟踪任意轨迹时，若存在一定的外部扰动，则可能会带来稳态误差，不能达到要求的性能指标。为了解决这一问题，T. L. Chern 等[108~112]提出了一种积分变结构控制方案，并且在伺服电机、机械臂等系统上得到了成功的应用。然而，常规积分变结构控制具有一定的局限性，它要求控制对象的系统模型是可控标准型，不包括任何零点。为了克服这一局限性，J. D. Wang 等[113]给出了另一种积分变结构控制方法，该方法成功地解决了对象的局限性，而且在满足匹配的条件下，该方法对于最小相位系统及非最小相位系统均适用。

1.5 滑模变结构控制应用

滑模变结构控制在机器人、航空航天和伺服系统领域有着广泛的应用[114]。在这些领域中，被控对象都存在严重的非线性，并且存在参数摄动或外界扰动和未建模动态。

1.5.1 在电机中的应用

滑模变结构控制最主要的应用领域是电机控制领域。变结构创始人之一 A. Utkin 等在著作[115]中详细探讨了变结构控制在变频器、直流电机、永磁同步电机和感应电机中的设计方法。

1.5.2 在机器人控制中的应用

由于机器人系统是典型的非线性系统，存在着多种不可预见的外部干扰，所以机器人控制是近年来滑模变结构控制理论的主要应用环境之一[116]。1983年，J. J. Slotine等[4]首次采用滑模控制方法设计了二自由度刚体机械手的滑模变结构控制器，实现了时变参考轨迹跟踪的控制。随后，国内外出现了大量的关于机器人滑模变结构控制的研究。Z. H. Man等[117]针对多关节刚性机器人设计了Terminal滑模控制器，使各关节按指定时间进行位置跟踪。A. Ficola等[118]采用滑模控制方法，通过设计两个滑动模面，实现了带有一个弹性力臂的两关节机器人控制。

1.5.3 在飞行器控制中的应用

滑模变结构控制的另一个应用环境是飞行器的运动控制。高为炳在著作[3]中分别针对航空航天飞行器、柔性空间飞行器的变结构控制进行了设计。C. Edwards等在著作[119]中探讨了L-1011型巡航飞行器的变结构控制方法。D. Zhou等[120]采用自适应变结构控制方法，设计了某型导弹的鲁棒控制律。由于滑模控制本身的优越性，使其很适合于导弹控制，近几年在这方面的研究成果有许多报道。近年来，基于滑模变结构的导弹制导控制发展很快，D. C. Liaw等[121]实现了导弹末制导的变结构控制，并有效地抑制了抖振；F. K. Yeh等[122]采用四元最优积分滑模变结构控制的方法，实现了中程巡航导弹的矢量非线性滑模控制，通过仿真进行了详细的分析，仿真中考虑了导弹转动惯量变化、空气动力学及阵风干扰的影响。周荻[123]将自适应滑模控制器和模糊滑模控制器应用于空间拦截控制。

1.5.4 在倒立摆控制中的应用

P. G. Grossimon等[124]通过设计一种新型滑动模面，使摆的角度成为基座转动角度的函数，从而实现了平面式倒立摆的滑模控制。Y. P. Chen等[125]采用滑模控制方法设计了两并行二级倒立摆的离散滑模控制器，并通过计算机实时控制实验，同时实现了小车位置、摆1角度及摆2角度的跟踪。张克勤等[126]利用倒立摆的特征值，设计了一种全鲁棒滑模面，实现了具有单输入的三级倒立摆全鲁棒滑模控制。

1.5.5 在伺服系统中的应用

复杂伺服系统具有非线性和不确定性，存在很多不利于系统性能提高的因素，如：

(1) 非线性因素：摩擦力矩、电机力矩波动、驱动饱和、耦合力矩、干扰力矩等；

(2) 参数变化：负载变化带来的转动惯量变化、温度升高导致的参数漂移等；

(3) 机械谐振及高频未建模动态；

(4) 测量延迟及测量噪声。

由于上述因素的存在，建立精确的数学模型是很困难的，只能建立一个近似的数学模

型。在建模时,要做合理的近似处理,要忽略对象中的不确定因素,诸如参数误差、未建模动态、测量噪声以及不确定的外干扰等。由近似模型出发设计控制器,设计中被忽略的不确定因素会引起控制系统品质的恶化,甚至导致不稳定。因此,考虑对象的不确定性,使所设计的控制器在不确定性对系统品质的破坏最严重时也能满足要求,具有一定的理论和工程实际意义。由于滑模变结构控制的特点,使它很适合于伺服系统的控制[130]。

参考文献

1. Utkin V I. Variable structure systems with sliding modes. IEEE Transactions Automatic Control, 1977, 22(2): 212~222
2. Young K D, Utkin V I, Ozguner U. A Control Engineer's Guide to Sliding Mode Control. IEEE Transactions on Control Systsms Technology, 1999, 7(3): 328~342
3. 高为炳. 变结构控制的理论及设计方法. 北京: 科学出版社, 1996
4. Slotine J J, Sastry S S. Tracking control of nonlinear systems using sliding surfaces with application to robot manipulator. International Journal of Control, 1983, 38(2): 465~492
5. Chung S C Y, Lin C L. A transformed Lure problem for sliding mode control and chattering reduction. IEEE Transactions on. Automatic Control, 1999, 44(3): 563~568
6. Xu J X, Lee T H, Pan Y J. On the sliding mode control for DC servo mechanisms in the presence of unmodeled dynamics. Mechatronics, 2003, 13: 755~770
7. Erbatur K, Kawamura A. Chattering elimination via fuzzy boundary layer tuning. IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the], 2002, 3: 2131~2136
8. Chen M S, Hwang Y R, Tomizuka M. A state-dependent boundary layer design for sliding mode control. IEEE Transactions on, Automatic Control, 2002, 47(10): 1677~1681
9. Vicente P V, Gerd H. Chattering-free sliding mode control for a class of nonlinear mechanical systems. International Journal of Robust and Nonlinear Control, 2001, 11: 1161~1178
10. Seshagiri S, Khalil H K. On introducing integral action in sliding mode control. Decision and Control, Proceedings of the 41st IEEE Conference on, 2002, 2: 1473~1478
11. 翟长连, 吴智铭. 一种离散时间系统的变结构控制方法. 上海交通大学学报, 2000, 34(5): 719~722
12. 于双和, 强文义, 傅佩琛. 无抖振离散准滑模控制. 控制与决策, 2001, 16(3): 380~382
13. Jiang K, Zhang J G, Chen Z M. A new approach for the sliding mode control based on fuzzy reaching law. Intelligent Control and Automation, Proceedings of the 4th World Congress on, 2002, 1: 656~660
14. Su W C, Drakunov S V, Ozguner U, Young K D. Sliding mode with chattering reduction in sampled data systems. Proceedings of the 32nd IEEE Conference on Decision and Control, 1993, 2452~2457
15. Kachroo P, Tomizuka M. Chattering reduction and error convergence in the sliding-mode control of a class of nonlinear systems. IEEE Transactions on Automatic Control, 1996, 41(7): 1063~1068
16. Kang B P, Ju J L. Sliding mode controller with filtered signal for robot manipulators using virtual plant/controller. Mechatronics, 1997, 7(3): 277~286
17. Yanada H, Ohnishi H. Frequency-shaped sliding mode control of an electrohydraulic servomoto. Journal of systems and Control and dynamics, 1999, 213(1): 441~448
18. Krupp D, Shtessel Y B. Chattering-free sliding mode control with unmodeled dynamics. American

Control Conference, 1999, 530～534

19. Xu J X, Pan Y J, Lee T H. A gain scheduled sliding mode control scheme using filtering techniques with applications to multi-link robotic manipulators. Journal of Dynamic Systems, Measurement, and Control, 2000, 122: 641～649
20. Kawamura A, Itoh H, Sakamoto K. Chattering reduction of disturbance observer based sliding mode control. IEEE Transactions on Industry Applications, 1994, 30(2): 456～461
21. Kim Y S, Han Y S, You W S. Disturbance observer with binary control theory. Power Electronics Specialists Conference, 1996, 2: 1229～1234
22. Liu H. Smooth sliding mode control of uncertain systems based on a prediction error. International Journal of Robust and Nonlinear Control, 1997, 7(4): 353～372
23. Eun Y, Kim J H, Kim K, Cho D H. Discrete-time variable structure controller with a decoupled disturbance compensator and its application to a CNC servomechanism. IEEE Transactions on Control Systems Technology, 1999, 7(4): 414～422
24. 宋立忠,陈少昌,姚琼荟. 多输入不确定系统离散变结构控制设计. 控制与决策,2003,18(4): 468～471
25. Bartolini G, Ferrara A, Usani E. Chattering avoidance by second-order sliding mode control. IEEE Transactions on Automatic Control, 1998, 43(2): 241～246
26. Bartolini G, Ferrara A, Usai E, Utkin V I. On multi-input chattering-free second-order sliding mode control. IEEE Transactions on Automatic Control, 2000, 1711～1717
27. Bartolini G, Punta E. Chattering elimination with second-order sliding modes robust to coulomb friction. Journal of Dynamic Systems, Measurement, and Control, 2000, 122: 679～686
28. Bartolini G, Pisano A, Punta E, Usai E. A Survey of applications of second-order sliding mode control to mechanical systems. International Journal of Control, 2003, 76(9): 875～892
29. Hamerlain M, Youssef T, Belhocine M. Switching on the derivative of control to reduce chatter. IEE Proceedings on Control Theory and Applications, 2001, 148(1): 88～96
30. 晁红敏,胡跃明. 动态滑模控制及其在移动机器人输出跟踪中的应用. 控制与决策,2001,16(5): 565～568
31. Q. P. Ha, Q. H. Nguyen, D. C. Rye, H. F. Durrant-Whyte, Fuzzy sliding-mode controllers with applications, IEEE Transactions on Industrial Electronics, 2001, 48(1): 38～41
32. Zhuang K Y, Su H Y, Chu J, Zhang K Q. Globally stable robust tracking of uncertain systems via fuzzy integral sliding mode control. Proceedings of the 3th World Congress on Intelligent Control and Automation, P. R. China, 2000, 1827～1831
33. Ryu S H, Park J H. Auto-tuning of sliding mode control parameters using fuzzy logic. American Control Conference, 2001, 618～623
34. 张天平,冯纯伯. 基于模糊逻辑的连续滑模控制. 控制与决策, 1995, 10(6): 503～507
35. 孙宜标,郭庆鼎,孙艳娜. 基于模糊自学习的交流直线伺服系统滑模变结构控制. 电工技术学报,2001, 16(1): 52～56
36. Morioka H, Wada K, Sabanovic A, Jezernik K. Neural network based chattering free sliding mode control. Proceedings of the 34th SICE Annual Conference, 1995, 1303～1308
37. Ertugrul M, Kaynak O. Neuro Sliding Mode Control of Robotic Manipulators. Mechatronics, 2000, 10(1,2): 239～263
38. Huang S J, Huang K S, Chiou K C. Development and application of a novel radial basis function sliding mode controller. Mechatronics, 2003, 13: 313～329
39. 达飞鹏,宋文忠. 基于输入输出模型的模糊神经网络滑模控制. 自动化学报,2000,26(1): 136～139

40. Ng K C, Li Y, Murray-Smith D J, Sharman K C. Genetic algorithms applied to fuzzy sliding mode controller design. Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. First International Conference on, 12-14 Sep 1995, 220～225
41. Lin F J, Chou W D. An induction motor servo drive using sliding-mode controller with genetic algorithm. Electric Power Systems Research, 2003, 64(2):93～108
42. Zhang C F, Wang Y N, He J, Long Y H. GA-NN-integrated sliding-mode control system and its application in the printing press. Control Theory & Applications, 2003, 20(2): 217～222
43. Hwang C L. Sliding mode control using time-varying switching gain and boundary layer for electrohydraulic position and differential pressure control. IEE Proceedings-Control Theory and Applications, 1996, 143(4), 325～332
44. Wong L J, Leung F H F, Tam P K S. A chattering elimination algorithm for sliding mode control of uncertain non-linear systems. Mechatronics, 1998, 8: 765～775
45. 林岩，毛剑琴，操云甫. 鲁棒低增益变结构模型参考自适应控制. 自动化学报, 2001, 27(5): 665～670
46. Lin F J, Wai R J. Sliding-mode-controlled slider-crank mechanism with fuzzy neural network. IEEE Transactions on Industrial Electronics, 2001, 48(1): 60～70
47. Xu J X, Lee T H, Wang M, Yu X H. Design of variable structure controllers with continuous switching control, International Journal of Control, 1996, 65(3): 409～431
48. Yang D Y, Yamane Y, Zhang X J, Zhu R Y. A new method for suppressing high-frequency chattering in sliding mode control system. Proceedings of the 36th SICE Annual Conference, 1997, 1285～1288
49. Konno Y, Hashmoto H. Design of sliding mode dynamics in frequency domain. IEEE Workshop on Variable Structure and Lyapunov Control of Uncertain Dynamical Systems, 1992, 120～125
50. Edwards C. A practical method for the design of sliding mode controllers using linear matrix inequalities. Automatica, 2004, 1～9
51. Sarpturk S Z, Istefanopulos Y, Kaynak O. On the stability of discrete-time sliding mode control system. IEEE Transactions on Automatic Control, 1987, 32(10): 930～932
52. Furuta K. Sliding mode control of a discrete system. Systems & Control Letters, 1990, 14(2): 145～152
53. Gao W B, Wang Y F, Homaifa A. Discrete-time variable structure control systems. IEEE Transactions on Industrial Electronics, 1995, 42(2): 117～122
54. Pan Y D, Furuta K. Discrete-time VSS Controller Design. International Journal of Robust and Nonlinear Control, 1997, 7(4): 373～386
55. Koshkouei A J, Zinober A S I. Sliding mode control of discrete-time systems. Journal of Dynamic Systems, Measurement, and Control, 2000, 122: 793～802
56. Kim J H, Oh S H, Cho D I, Hedrick J K. Robust discrete-time variable structure control methods. Journal of Dynamic Systems, Measurement, and Control, 2000, 122: 766～775
57. Emelyanov S V, Korovin S K, Mamedov I G. Variable Structure Control Systems: Discrete and Digital. Mir Publishers, Moscow, 2000
58. Sira R H, Llanes S O. Adaptive dynamical sliding mode control via backstepping. Proceedings of the 32nd IEEE Conference on Decision and Control, 1993, 2: 1422～1427
59. Bolivar M R, Zinober A S I, Sira-Ramirez H. Dynamic Adaptive sliding mode output tracking control of a class of nonlinera systems. International Journal of Robust and Nonlinear Control, 1997, 7(4): 387～405
60. Lin F J, Chiu S L, Shyu K K. Novel sliding mode controller for synchronous motor drive. IEEE

Transactions on Aerospace and Electronic Systems, 1998, 34(2): 532~542

61. Wheeler G, Su C H, Stepanenko Y. A sliding mode controller with improved adaptation laws for the upper bounds on the norm of uncertainties. Automatica, 1998, 34(12): 1657~1661
62. Bekiroglu N, Bozma H I, Istefanopulos Y. Model reference adaptive approach to sliding mode control. American Control Conference, 1995,1:1028~1032
63. Song J B, Ishida Y. A Robust Sliding Mode Control for Pneumatic Servo Systems. International Journal Engineering Science, 1997, 35(8): 711~723
64. Kwan C M. Sliding mode control of linear systems with mismatched uncertainties. Automatica, 1995, 31(2): 303~307
65. Choi H H. A new method for variable structure control system design: A linear matrix inequality approach. Automatica, 1997. 33(11): 2089~2092
66. Choi H H. An explicit formula of linear sliding surface for a class of uncertain dynamic systems with mismatched uncertainties. Automatica. 1998, 34(8): 1015~1020
67. Choi H H. On the existence of linear sliding surface for a class of uncertain dynamic systems with mismatched uncertainties. Automatica, 1999, 35: 1707~1715
68. Li J. Backstepping variable structure control of nonlinear systems with unmatched uncertainties, 14th Triennial World Congress, Beijing, China, 1999, 67~71
69. Koshkouei A J, Zinober A S I. Adaptive backstepping control of nonlinear systems with unmatched uncertainty. Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia, 2000, 4765~4770
70. Gouaisbaut F, Perruqetti W, Richard J P. A sliding mode control for linear systems with input and state delays. Proceedings of the 38th IEEE Conference on Decision and Control, 1999,4: 4234~4239
71. Chou C H, Cheng C C. Decentralized mode following variable structure control for perturbed large scale systems with time-delay interconnections. Proc. American Control Conference, 2000, 641~645
72. Xia Y Q, Yingmin Jia. Robust sliding-mode control for uncertain time-delay systems: a LMI approach. Automatic Control, IEEE Transactions on, 2003,48(6):1086~1091
73. Shtessel Y B, Zinober A S I, Shkolnikov I A. Sliding mode control for nonlinear systems with output delay via method of stable center. Journal of Dynamic Systems, Measurement, and Control, 2003,125: 158~165
74. Utkin V I. Sliding modes in control optimization. Berlin: Spring-Verlag, 1992
75. Chang Y F, Chen B S. A robust performance variable structure PI/P control design for high precise positioning control systems. International Journal of Machine Tools and Manufacture, 1995, 35(12): 1649~1667
76. Lu X Y, Spurgeon S. K. Control of nonlinear non-minimum phase systems using dynamic sliding mode. Int. J. System Science, 1999, 30(2): 183~198
77. Wang J, Zheng Y, Lu X P. Robust output tracking of constrained nonlinear systems. 14th Triennial World Congress, Beijing, China, 1999, 37~40
78. Bartolini G, Ferrara A, Giacomini L. Modular backstepping design of an estimation-based sliding mode controller for uncertain nonlinear plants. Proceedings of the 1998 American Control Conference, 1998, 1:574~578
79. Yu X H, Xu J X. Variable structure systems: toward the 21st Century. Springer, Berlin, 2002
80. Yu X H, Man Z H. Model refrence adaptive control systems with terminal sliding modes. International Journal of Control, 1996, 64(6): 1165~1176

81. Man Z H, Yu X H. Terminal sliding mode control of MIMO linear systems. Proceedings of the 35th Conference on Decision and Control, 1996,4619～4624
82. Wu Y, Yu X H, Man Z H. Terminal sliding mode control design for uncertain dynamic systems. Systems and Control Letters, 1998, 34: 281～288
83. Feng Y, Yu X H, Man Z H. Non-singular terminal sliding mode control of rigid manipulators. Automatica, 2002, 38: 2159～2167
84. Tao C W, Taur J S, Chan M L. Adaptive fuzzy terminal sliding mode controller for linear systems with mismatched time-varying uncertainties. IEEE Transactions on Systems,2003,Part B:1～8
85. Park K B, Teruo T. Terminal sliding mode control of second-order nonlinear uncertain systems. International Journal of Robust and Nonlinear Control, 1999,9:769～780
86. 庄开宇,张克勤,苏宏业,褚健.高阶非线性系统的 Terminal 滑模控制.浙江大学学报,2002,36(5):482～485
87. Zhuang K Y, Su H Y, Zhang K Q, Chu J. Adaptive terminal sliding mode control for high order nonlinear dynamic systems. 2003, 4(1):58～63
88. Lu Y S, Chen J S. Design of a global sliding-mode controller for a motor drive with bounded control. International Journal of Control, 1995, 62(5): 1001～1019
89. Choi H S, Park Y H, Cho Y S, Lee M H. Global Sliding Mode Control. IEEE Control Systems Magazine, 2001,21(3): 27～35
90. 肖雁鸿,葛召炎,周靖林,彭永劲.全滑模变结构控制系统.电机与控制学报,2002,6(3): 233～236
91. 米阳,李文林,井元伟,刘小平.线性多变量离散系统全程滑模变结构控制.控制与决策,2003,18(4):460～464
92. Yu X H, Xu J X. Variable structure systems: Towards the 21th Century. Spring-Verlag, Berlin Heidelberg, New York, 2002
93. Zheng Y H, Fattah H A A, Loparo K A. Non-linear adaptive sliding mode observer-controller scheme for induction motors, International Journal of Adaptive Control and Signal Processing, 2000, 14, Issue 2,3, 245～273
94. Tan C P, Edwards C. Sliding mode observers for robust detection and reconstruction of actuator and sensor faults. International Journal of Robust and Nonlinear Control, 2003, 13(5): 443～463
95. Kim S M, Han W Y, Kim S J. Design of a new adaptive sliding mode observer for sensorless induction motor drive. Electric Power Systems Research, 2004, 70: 16～22
96. Lee S S, Park J K. Design of power system stabilizer using observer/sliding mode, observer/sliding mode-model following and H_∞/sliding mode controllers for small-signal stability study. International Journal of Electrical Power & Energy Systems, 1998, 20(8):543～553
97. Lin S C, Chen Y Y. RBF network based sliding mode control. IEEE International Conference on Systems, Man, and Cybernetics, 1994, 1957～1961
98. Munoz D, Sbarbaro D. An adaptive sliding-mode controller for discrete nonlinear systems. IEEE Transactions on Industrial Electronics, 2000, 47(3): 574～581
99. Z. H. Man, X. H. Yu, K. Eshraghian, M. Palaniswami, A robust adaptive sliding mode tracking control using an RBF neural network for robotic manipulators, IEEE International Conference on Neural Networks, 1995, 5:2403～2408
100. J. M. Carrasco, J. M. Quero, F. P. Ridao, M. A. Perales, L. G. Franquelo, Sliding mode control of a DC/DC PWM converter with PFC implemented by neural networks, IEEE Transactions on Circuits and Systems- Ⅰ: Fundamental Theory and Applications, 1997, 44(8): 743～749

101. Parma G, Menezes B R, Braga A P, Costa M A. Sliding mode neural network control of an induction motor drive. International Journal of Adaptive Control and Signal Processing, 2003, 17(6):501~508

102. Kim S W, Lee J J. Design of a fuzzy controller with fuzzy sliding surface. Fuzzy Sets and Systems, 1995, 71(3): 359~367

103. Yoo B, Ham W. Adaptive fuzzy sliding mode control of nonlinear system. IEEE Trans. On Fuzzy Systems, 1998, 6(2): 315~321

104. Lu Y S, Chen J S. A self-organizing fuzzy sliding-mode controller design for a class of nonlinear servo systems. IEEE Transactions on Industrial Electronics, 1994, 41(5): 492~502

105. Liang C Y, Su J P. A new approach to the design of a fuzzy sliding mode controller. Fuzzy Sets and Systems, 2003, 139: 111~124

106. Wai R J, Lin C M, Hsu C F. Adaptive fuzzy sliding-mode control for electrical servo drive. Fuzzy Sets and Systems, 2004, 143: 295~310

107. Chen J Y. Expert SMC-based fuzzy control with genetic algorithms. Journal of the Franklin Institute, 1999, 336: 589~610

108. Chern T L, Wu Y C. Design of integral variable structure controller and application to electrohydraulic velocity servosystems. IEE Proceedings-D, 1991, 138(5): 439~444

109. Chern T L, Wu Y C. Integral variable structure control approach for robot manipulators. IEE Proceedings-D, 1992, 139(2): 161~166

110. Chern T L, Wu Y C. An optimal variable structure control with integral compensation for electrohydraulic position servo control systems. IEEE Transactions on industrial electronics, Proceedings-D, 1992, 39(5): 460~463

111. Chern T L, Wu Y C. Design of brushless DC position servo systems using integral variable structure approach. IEE Proceedings-B, 1993, 140(1): 27~34

112. Chern T L, Wong J S. DSP based integral variable structure control for motor servo drives. IEE Proc. Control Theory Appl. , 1995, 142(5): 444~450

113. Wang J D, Lee T L, Juang Y T. New methods to design an integral variable structure controller. IEEE Trans, on Automatic Control, 1996, 41(1): 140~143

114. Perruquetti W, Barbot J P. Sliding mode control in engineering. Marcel Dekker Inc. , New York, 2002

115. Utkin A, Guldner J, Shi J X. Sliding Mode Control in Electromechanical Systems. Taylor&Francis, 1999

116. Yu X H, Xu J X. Advances in Variable Structure Systems. World Scientific Publishing, Singapore, 2000

117. Man Z H, Paplinski A P, Wu H R. A robust MIMO terminal sliding mode control scheme for rigid robot manipulators. IEEE Transactions on Automatic Control, 1994, 39: 2464~2469

118. Ficola A, Cava M L. A sliding mode controller for a two-joint robot with an elastic link. Mathematics and Computers in Simulation, 1996, 41: 559~569

119. Edwards C, Spurgeon S K. Sliding Mode Control, Theory and Applications. Taylor&Francis, 1998

120. Zhou D, Mu C D, Xu W L. Adaptive sliding mode guidance of a homing missile. Journal of guidance, control and dynamics, 1999, 22(4): 589~592

121. Liaw D C, Liang Y W, Cheng C C. Nonlinear Control for Missile Terminal Guidance. Journal of Dynamic Systems, Measurement, and Control. 2000, 122: 663~668

122. Yeh F K, Chien H H, Fu L C. Design of optimal midcourse guidance sliding-mode control for missiles

with TVC. Aerospace and Electronic Systems, IEEE Transactions on, 2003, 39(3): 824～837

123. 周荻. 寻的导弹新型导引规律. 北京: 国防工业出版社, 2002

124. Grossimon P G, Barbieri E, Drakunov S. Sliding mode control of an inverted pendulum. Proceedings of the Twenty-Eighth Southeastern Symposium on System theory, 1996, 248～252

125. Chen Y P, Chang J L, Chu S R. PC-based sliding-mode control applied to parallel-type double inverted pendulum system, Mechatronics, 1999, 9: 553～564

126. 张克勤, 苏宏业, 庄开宇, 褚健. 三级倒立摆系统基于滑模的鲁棒控制. 浙江大学学报, 2002, 36(4): 401～409

127. Liu J K, Er L J. Sliding Mode Controller Design for Position and Speed Control of Flight Simulator Servo System with Large Friction. Systems Engineering and Electronics(China), 2003, 14(3): 59～62

128. 薛定宇. 控制系统计算机辅助设计[M]. 北京: 清华大学出版社, 1996

129. 刘金琨. 先进 PID 控制及其 MATLAB 仿真(第 2 版). 北京: 电子工业出版社, 2004

130. 姚琼荟, 黄继起, 吴汉松. 变结构控制系统. 重庆: 重庆大学出版社, 1997

131. 王丰尧. 滑模变结构控制. 北京: 机械工业出版社, 1995

132. Emelyanov S V, Korovin S K, Mamedov I G. Variable structure control systems: discrete and digital. Mir, Moscow, 1995

133. 胡跃明. 变结构控制理论与应用. 北京: 科学出版社, 2003

134. Utkin V I, Guldner J, Shi J. Sliding mode control in electromechanical systems. Taylor & Francis, 1999

第2章　连续时间系统滑模控制

2.1　滑动模态的存在和到达条件

滑动模态存在条件的成立是滑动模态控制应用的前提。如果系统的初始点 $x(0)$ 不在 $s=0$ 附近，而是在状态空间的任意位置，此时要求系统的运动必须趋向于切换面 $s=0$，即必须满足可达性条件，否则系统无法启动滑模运动。由于滑模变结构控制的控制策略多种多样，对于系统可达性条件的实现形式也不尽相同，滑动模态存在的数学表达式为

$$\lim_{s\to 0^+}\dot{s}<0,\qquad \lim_{s\to 0^-}\dot{s}>0 \tag{2.1}$$

式(2.1)意味着在切换面邻域内，运动轨线将于有限时间内到达切换面，所以也称为局部到达条件。到达条件的等价形式为

$$s\dot{s}<0 \tag{2.2}$$

其中切换函数 $s(x)$ 应满足以下条件：

(1) 可微；

(2) 过原点，即 $s(0)=0$。

由于状态 x 可以取任意值，即 x 离开切换面可以任意远，故到达条件(2.2)也称为全局到达条件。为了保证在有限时刻到达，避免渐进趋近，可对式(2.2)进行修正：

$$s\dot{s}<-\delta \tag{2.3}$$

其中 $\delta>0$，δ 可以取得任意小。

通常将式(2.2)表达成李雅普诺夫函数型的到达条件：

$$\dot{V}(x)<0,\qquad V(x)=\frac{1}{2}s^2 \tag{2.4}$$

其中 $V(x)$ 为定义的李雅普诺夫函数。

2.2　等效控制及滑动模态方程

2.2.1　等效控制

设系统的状态方程为

$$\dot{x}=f(x,u,t)\qquad x\in\mathbf{R}^n,u\in\mathbf{R} \tag{2.5}$$

其中 u 为控制输入，t 为时间。

如果达到理想的滑动模态控制，则 $\dot{s}=0$，即

$$\dot{s}=\frac{\partial s}{\partial x}\frac{\partial x}{\partial t}=0\quad \text{或}\quad \frac{\partial s}{\partial x}f(x,u,t)=0 \tag{2.6}$$

将式(2.6)中 u 的解 u_{eq}(如果存在)称为系统在滑动模态区内的等效控制。等效控制往往是针对确定性系统在无外加干扰情况下进行设计的。

例如,对于线性系统

$$\dot{x} = \boldsymbol{A}x + \boldsymbol{b}u \qquad x \in \mathbf{R}^n, u \in \mathbf{R} \tag{2.7}$$

取切换函数

$$s(x) = \boldsymbol{c}x = \sum_{i=1}^{n} c_i x_i = \sum_{i=1}^{n-1} c_i x_i + x_n \tag{2.8}$$

其中 $x_i = x^{(i-1)}(i=1,2,\cdots,n)$为系统状态及其各阶导数,选取常数 $c_1,c_2,\cdots,c_{n-1}$,使得多项式 $p^{n-1}+c_{n-1}p^{n-2}+\cdots+c_2p+c_1$ 为 Hurwitz 稳定,p 为 Laplace 算子。

设系统进入滑动模态后的等效控制为 u_{eq},由式(2.7)有

$$\dot{s} = \boldsymbol{c}\dot{x} = \boldsymbol{c}(\boldsymbol{A}x + \boldsymbol{b}u_{eq}) = 0 \tag{2.9}$$

若矩阵$[\boldsymbol{cb}]$满秩,则可解出等效控制

$$u_{eq} = -[\boldsymbol{cb}]^{-1}\boldsymbol{cA}x \tag{2.10}$$

针对带有不确定性和外加干扰的系统,一般采用的控制律为等效控制加切换控制,即

$$u = u_{eq} + u_{vss} \tag{2.11}$$

其中切换控制 u_{vss} 实现对不确定性和外加干扰的鲁棒控制。所设计的控制律 u 需要满足滑模稳定条件。

2.2.2 滑动模态运动方程

有了等效控制后,可写出滑动模态运动方程。将等效控制 u_{eq} 代入系统的状态方程(2.5),可得

$$\begin{gathered}\dot{x} = f(x,u_{eq},t) \qquad x \in \mathbf{R}^n, u \in \mathbf{R} \\ s(x) = 0\end{gathered} \tag{2.12}$$

将式(2.10)代入式(2.7)所示的线性系统,有

$$\begin{cases}\dot{x} = [\boldsymbol{I} - \boldsymbol{b}(\boldsymbol{cb})^{-1}\boldsymbol{c}]\boldsymbol{A}x \\ s(x) = \boldsymbol{c}x = 0\end{cases} \tag{2.13}$$

式中 $\boldsymbol{I}$ 为单位阵。

滑动模态运动是系统沿切换面 $s(x)=0$ 上的运动,到达理想终点时,满足 $s=0$ 及 $\dot{s}=0$,同时切换开关必须是理想开关,这是一种理想的极限情况。实际上,系统运动点沿切换面上下穿行。所以式(2.13)是滑模变结构控制系统在滑动模态附近的平均运动方程,这种平均运动方程描述了系统在滑动模态下的主要动态特性。通常希望这个动态特性既是渐近稳定的,又具有优良的动态品质。从式(2.13)中可以看出,滑动模态运动的渐近稳定性和动态品质取决于切换函数 s 及其参数的选择。

2.3 滑模变结构控制匹配条件及不变性

滑模变结构控制的突出优点是可以实现滑动模态与系统的外干扰和参数摄动完全无关,这种性质称为滑动模态的不变性,这也是滑模变结构控制受到重视的主要原因。但对于

一般线性系统,不变性的成立是有条件的,需要满足滑动模态匹配条件。分以下三种情况进行讨论。

(1) 系统受外干扰时

$$\dot{x} = Ax + Bu + Df \tag{2.14}$$

其中 Df 表示系统所受的外干扰。

滑动模态不受干扰 f 影响的充分必要条件为

$$\operatorname{rank}[\boldsymbol{B},\boldsymbol{D}] = \operatorname{rank}\boldsymbol{B} \tag{2.15}$$

如果式(2.15)满足,则系统可化为

$$\dot{x} = Ax + B(u + \tilde{D}f) \tag{2.16}$$

其中 $\tilde{\boldsymbol{D}}=\boldsymbol{B}^{-1}\boldsymbol{D}$,则通过设计控制律 u 可实现对干扰的完全补偿。条件式(2.15)称为干扰和系统的完全匹配条件。

(2) 系统存在不确定性时

$$\dot{x} = Ax + Bu + \Delta Ax \tag{2.17}$$

滑动模态与 $\Delta\boldsymbol{A}$ 不确定性无关的充分必要条件为

$$\operatorname{rank}[\boldsymbol{B},\Delta\boldsymbol{A}] = \operatorname{rank}\boldsymbol{B} \tag{2.18}$$

如果式(2.18)满足,则系统可化为

$$\dot{x} = Ax + B(u + \Delta\tilde{A}x) \tag{2.19}$$

其中 $\Delta\boldsymbol{A}=\boldsymbol{B}\Delta\tilde{\boldsymbol{A}}$,则通过设计控制律 u 可实现对不确定性的完全补偿。条件式(2.18)称为不确定性和系统的完全匹配条件。

(3) 对于同时存在外干扰和参数摄动的系统

$$\dot{x} = Ax + \Delta Ax + B + Df \tag{2.20}$$

如满足匹配条件式(2.15)和式(2.18),则系统可化为

$$\dot{x} = Ax + B(u + \Delta\tilde{A}x + \tilde{D}f) \tag{2.21}$$

2.4 滑模控制器设计基本方法

设计滑模变结构控制器的基本步骤包括两个相对独立的部分:

(1) 设计切换函数 $s(\boldsymbol{x})$,使它所确定的滑动模态渐近稳定且具有良好的动态品质;

(2) 设计滑动模态控制律 $u^{\pm}(\boldsymbol{x})$,使到达条件得到满足,从而在切换面上形成滑动模态区。

一旦切换函数 $s(\boldsymbol{x})$和滑动模态控制律 $u^{\pm}(\boldsymbol{x})$都得到了,滑动模态控制系统就能完全建立起来。

常规滑模变结构控制有以下几种设计方法:

(1) 常值切换控制

$$u = u_0\operatorname{sgn}(s(\boldsymbol{x})) \tag{2.22}$$

式中,u_0 是待求的常数,sgn 是符号函数,求滑模变结构控制就是求 u_0。

(2) 函数切换控制

$$u = u_{eq} + u_0\operatorname{sgn}(s(\boldsymbol{x})) \tag{2.23}$$

这是以等效控制 u_{eq} 为基础的形式。

(3) 比例切换控制

$$u = \sum_{i=1}^{k} \psi_i x_i \qquad k < n$$

$$\psi_i = \begin{cases} \alpha_i & x_i s < 0 \\ \beta_i & x_i s > 0 \end{cases} \quad \alpha_i, \beta_i \text{ 为常数} \tag{2.24}$$

2.5 基于比例切换函数的滑模控制

2.5.1 控制器设计方法

设位置状态方程为

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u \tag{2.25}$$

设位置指令信号为 r,将系统的位置误差 e 和速度误差 $\dot{e}$ 作为状态变量,即

$$\left.\begin{aligned} e &= r - x(1) \\ \dot{e} &= \dot{r} - x(2) \end{aligned}\right\} \tag{2.26}$$

则切换函数为

$$s = ce + \dot{e} \tag{2.27}$$

根据比例切换控制方法,控制律取为

$$u = (\alpha \mid e \mid + \beta \dot{e})\mathrm{sgn}(s) \tag{2.28}$$

其中 α 和 β 为大于零的常数。

2.5.2 仿真实例

考虑如下时变对象:

$$G_p(s) = \frac{b}{s^2 + as} \tag{2.29}$$

其中 $a=25+5\sin(6\pi t)$,$b=133+50\sin(2\pi t)$。

将传递函数描述为状态方程的形式:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u \tag{2.30}$$

其中 $\boldsymbol{A}=\begin{bmatrix} 0 & 1 \\ 0 & -a \end{bmatrix}$,$\boldsymbol{B}=\begin{bmatrix} 0 \\ b \end{bmatrix}$。

采用基于比例的切换函数控制方法,S 1 为阶跃响应,$S=2$ 为正弦响应。在控制律中,取 $c=30$,$\alpha=500$,$\beta=10$。

1. 仿真方法之一

采用 M 函数设计控制器,针对时变对象式(2.30)进行仿真,取 S=1,仿真结果如图 2-1~图 2-3 所示。

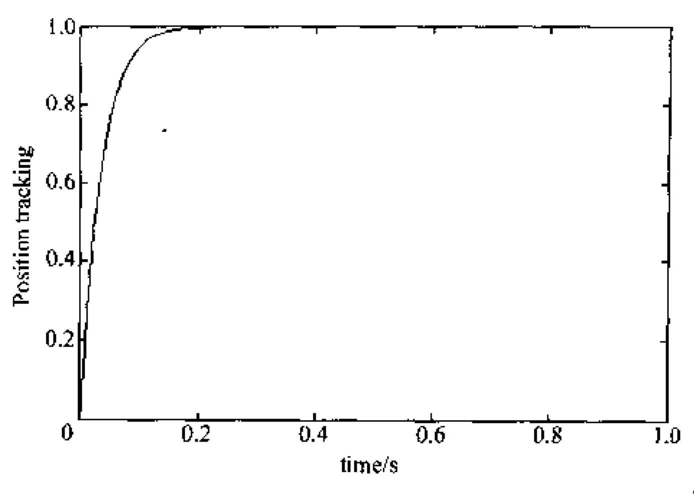


图 2-1 阶跃响应

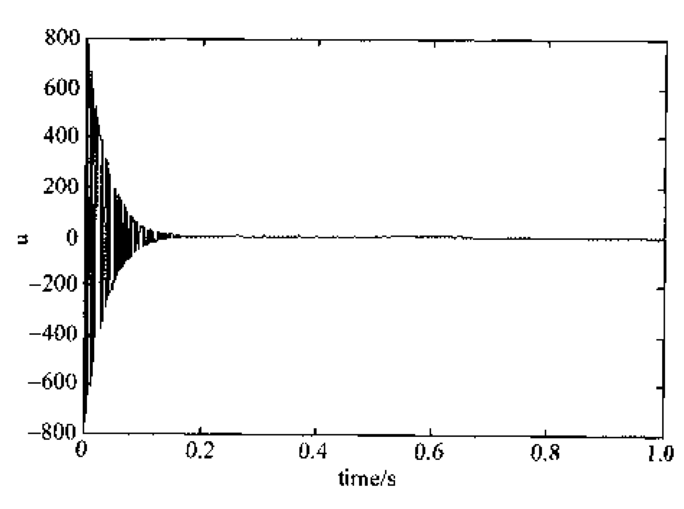


图 2-2 控制器输出

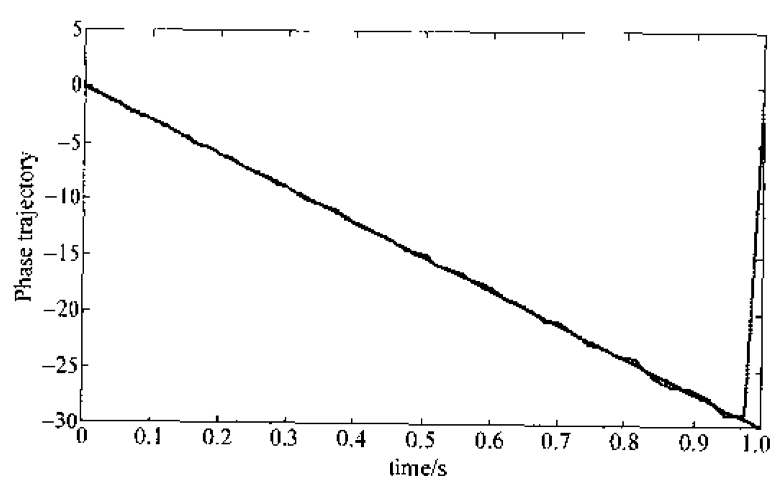


图 2-3 滑模运动相轨迹

主程序：chap2_1.m

```
clear all;
```

```
close all;
global S A F c alfa beta

xk = [0,0];

ts = 0.001;
T = 1;
TimeSet = [0:ts:T];

[t,y] = ode45('chap2_1eq',TimeSet,xk,[],[]);
x1 = y(:,1);
x2 = y(:,2);

if S == 1
   rin = 1.0;
   drin = 0;
elseif S == 2
   rin = A * sin(F * 2 * pi * t);
   drin = A * F * 2 * pi * cos(F * 2 * pi * t);
end

e1 = rin - x1;
e2 = drin - x2;
s = c * e1 + e2;

for k = 1:1:T/ts + 1
   u(k) = (alfa * abs(e1(k)) + beta * abs(e2(k))) * sign(s(k));
end

figure(1);
plot(t,rin,'r',t,y(:,1),'b');
xlabel('time(s)');ylabel('Position tracking');
figure(2);
plot(t,u,'r');
xlabel('time(s)');ylabel('u');
figure(3);
plot(e1,e2,'r',e1, - c * e1,'b');
xlabel('time(s)');ylabel('Phase trajectory');
```

控制子程序：chap2_1eq.m

```
function dx = PlantModel(t,x,flag,para)
global S A F c alfa beta
dx = zeros(2,1);

S = 1;
if S == 1
   rin = 1.0;
   drin = 0;
elseif S == 2
   A = 0.5;F = 3;
```

```
    r = 0.5*sin(3*2*pi*t);
    dr = 0.5*3*2*pi*cos(3*2*pi*t);
end

c=15;
alfa=500;
beta=10;

e=r-x(1);
de=dr-x(2);

s=c*e+de;
u=(alfa*abs(e)+beta*abs(de))*sign(s);

dx(1)=x(2);
dx(2)=-(25+5*sin(3*2*pi*t))*x(2)+(133+50*sin(1*2*pi*t))*u;
```

2. 仿真方法之二

采用Simulink进行仿真。被控对象选取线性时变对象式(2.30)。当指令取幅值为0.50,频率为3Hz的正弦信号时,仿真结果如图2-4所示;当指令取幅值为0.50,频率为1.0Hz的方波信号时,仿真结果如图2-5所示。



图2-4　正弦跟踪

仿真程序:仿真主程序为chap2_2.mdl,如图2-6所示。其中时变对象和控制器采用S函数的形式,时变对象的Simulink模块如图2-7所示,控制器的Simulink模块如图2-8所示。

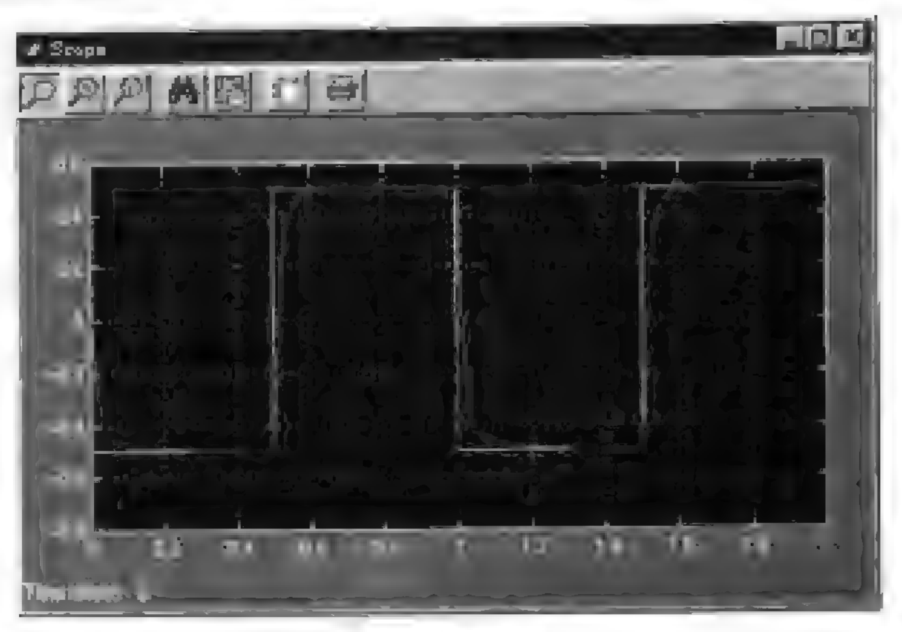


图2-5 方波响应

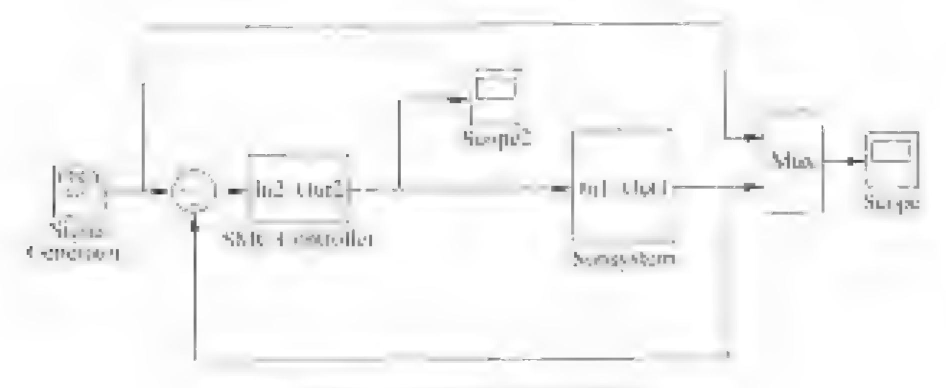


图2-6 Simulink仿真主程序

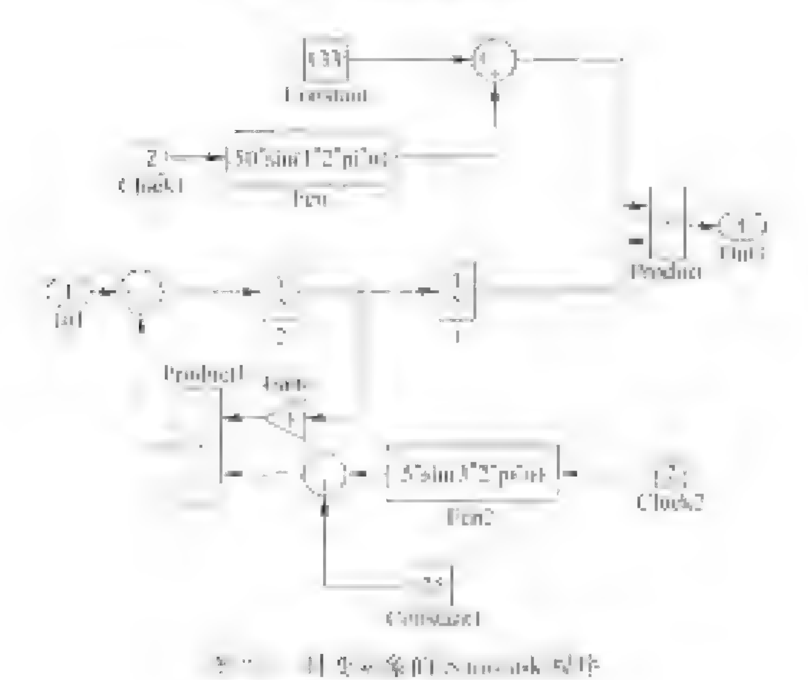


图2-7 被控对象的Simulink程序

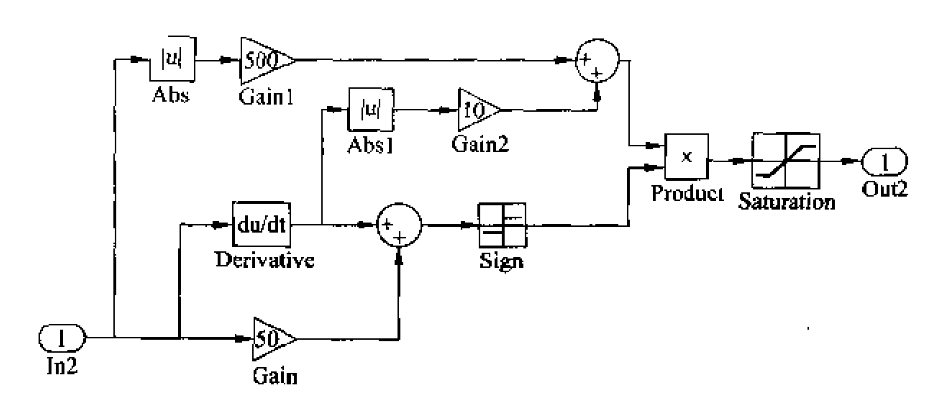


图 2-8 控制器的 Simulink 程序

2.6 台车式倒立摆的滑模控制

2.6.1 台车式倒立摆模型

倒立摆系统的控制问题一直是控制研究中的一个典型问题。控制的目标是通过给小车底座施加一个控制量，使小车停留在预定的位置，并使杆不倒下，即不超过一预先定义好的垂直偏离角度范围。图 2-9 为一级倒立摆的示意图，小车质量为 M，摆的质量为 m，小车位置为 x，摆的角度为 α，杆长为 $2a$。

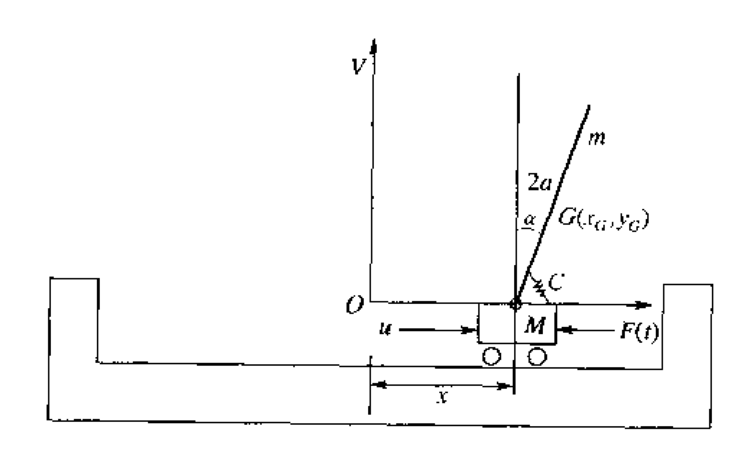


图 2-9 倒立摆系统示意图

倒立摆的状态方程为：

$$\dot{x} = \boldsymbol{A}x + \boldsymbol{B}(u + f(t)) \tag{2.31}$$

其中

$$A=\begin{bmatrix}0&0&1&0\\0&0&0&1\\0&a_{32}&0&0\\0&a_{42}&0&0\end{bmatrix},\quad B=\begin{bmatrix}0\\0\\b_3\\b_4\end{bmatrix},\quad x=\begin{bmatrix}x\\\alpha\\\dot{x}\\\dot{\alpha}\end{bmatrix}$$

且

$$a_{32}=-3(C_1-mga)/[a(4M+m)]$$
$$a_{42}=-3(M+m)(C_1-mga)/[a^2m(4M+m)]$$
$$b_3=4/(4M+m)$$
$$b_4=3/(4M+m)$$

其中 C_1 为弹性硬度，$g=9.8\text{m/s}^2$，u 和 $f(t)$ 分别为控制输入和干扰力，且 $|f(t)|\leqslant f_0$，f_0 为常数。

2.6.2 滑模控制器设计

V. I. Utkin[1] 提出并采用了两种变结构控制方法实现了台车式倒立摆小车和摆角的控制。在该控制律中，采用 Ackermann 公式设计滑模控制律中的 $\boldsymbol{C}$ 值：

$$s=\boldsymbol{C}^{\mathrm{T}}\boldsymbol{x},\qquad \boldsymbol{C}^{\mathrm{T}}=\boldsymbol{e}^{\mathrm{T}}\boldsymbol{P}(\boldsymbol{A}) \tag{2.32}$$

Ackermann 公式描述为

$$\boldsymbol{e}^{\mathrm{T}}=[0,\cdots,0,1][\boldsymbol{b},\boldsymbol{Ab},\cdots,\boldsymbol{A}^{n-1}\boldsymbol{b}]^{-1} \tag{2.33}$$

$$\boldsymbol{P}(\lambda)=(\lambda-\lambda_1)(\lambda-\lambda_2)\cdots(\lambda-\lambda_{n-1})(\lambda-\lambda_n) \tag{2.34}$$

(1) 常值切换控制

根据式(2.22)，控制律取

$$u=-M_0\,\mathrm{sgn}(s) \tag{2.35}$$

其中 $M_0>f_0$。

(2) 比例切换控制

根据式(2.24)，控制律取

$$u=-\beta(|\boldsymbol{x}|+|\dot{\boldsymbol{x}}|+|\boldsymbol{\alpha}|+|\dot{\boldsymbol{\alpha}}|)\mathrm{sgn}(s) \tag{2.36}$$

2.6.3 仿真实例

倒立摆的参数取为：$g=9.8\text{m/s}^2$(重力加速度)，$M=5.0\text{kg}$(小车质量)，$m=1\text{kg}$(杆的质量)，$a=0.5\text{m}$(杆的半长)，$C_1=1$，$f(t)=0.5\sin(3t)$。

作图采样时间为 $T=0.02$，仿真时间为 30。初始条件取 $\boldsymbol{x}(0)=0.5$，$\boldsymbol{\alpha}(0)=0.3$，$\dot{\boldsymbol{x}}(0)=0.0$，$\dot{\boldsymbol{\alpha}}(0)=0$。期望状态为：$\boldsymbol{\alpha}(0)=0$，$\dot{\boldsymbol{\alpha}}(0)=0$，$\boldsymbol{x}(0)=0$，$\dot{\boldsymbol{x}}(0)=0$，其中摆动角度单位为弧度。

仿真程序由两部分组成：主程序 chap2_3.m 和子程序 chap2_3eq.m。取 $F=1$，控制律采用式(2.35)，其中 $M_0=40$，仿真结果如图 2-10～图 2-12 所示；取 $F=2$，控制律采用式(2.36)，其中 $\beta=30$，仿真结果如图 2-13～图 2-15 所示。

在控制系统中，由于 $n=4$，故取 $\lambda=-1$，$\lambda_2=-2$，$\lambda_3=-3$，根据 Ackermann 公式设计

滑模控制律中的C值，即根据式(2.22)，可得C = [-5.2672 -15.[illegible] [illegible] 5.[illegible] 18.6334]

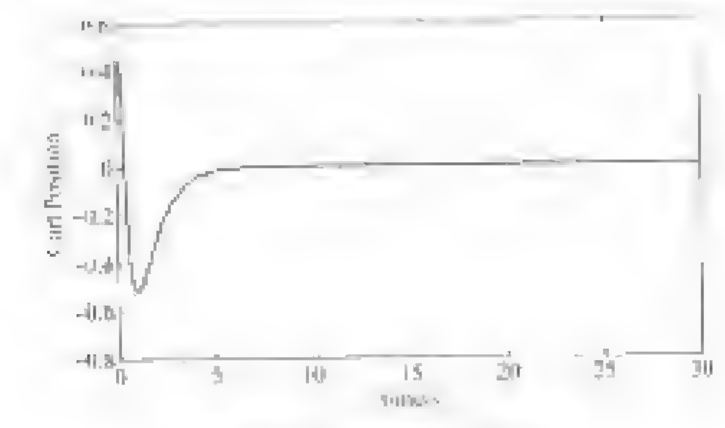


图2-10 小车位置控制(F=1)

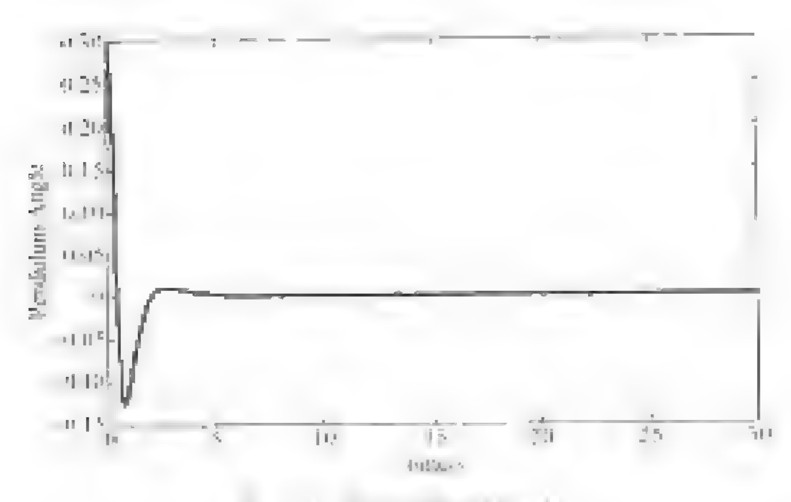


图2-11 摆角角度控制(F=1)

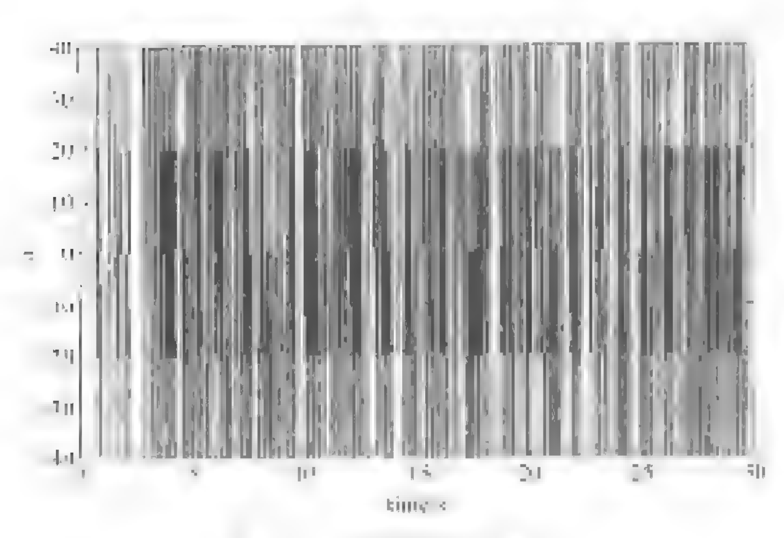


图2-12 控制器输出(F=1)

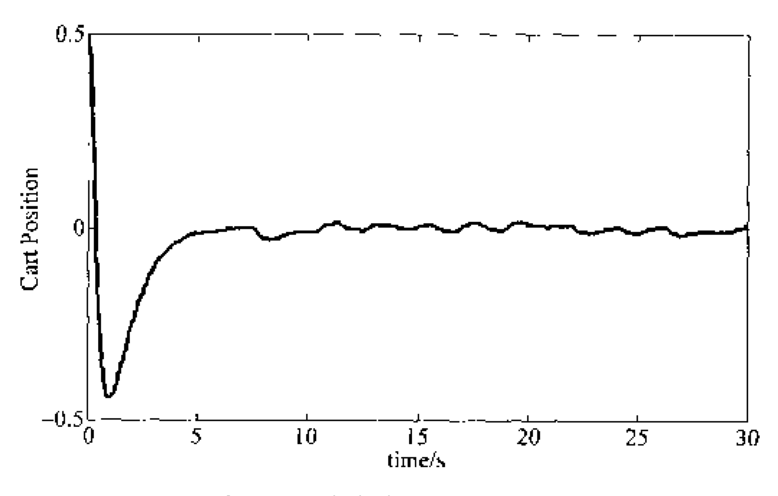


图 2-13　小车位置控制(F=2)

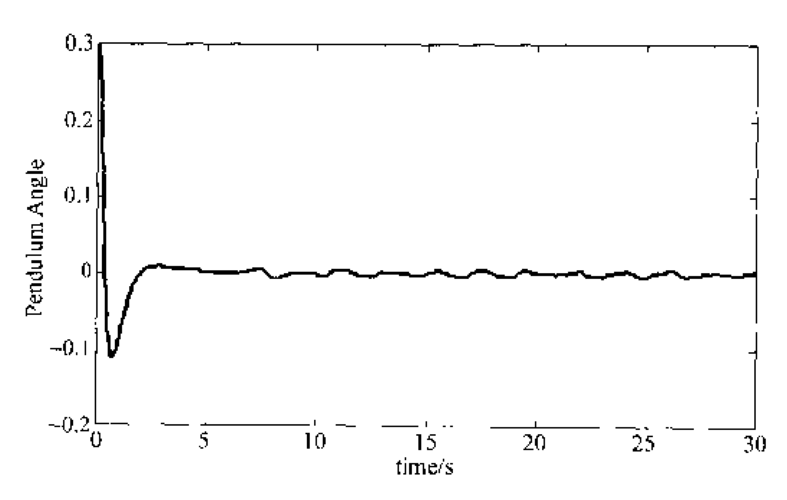


图 2-14　摆角度控制(F=2)

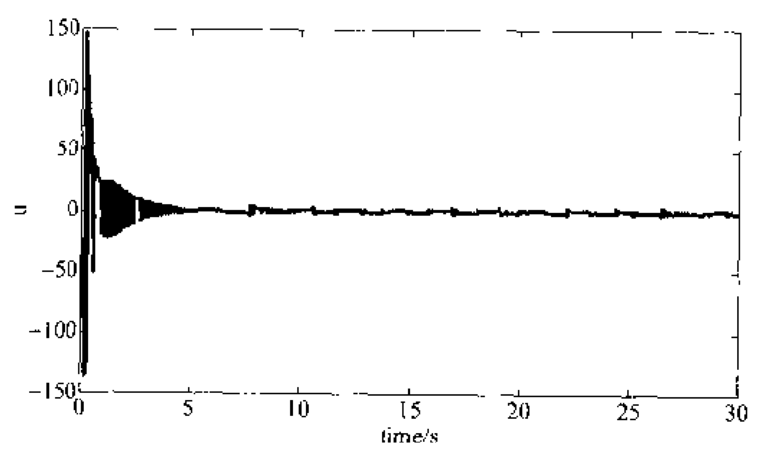


图 2-15　控制器输出(F=2)

(1) 主程序：chap2_3.m

```
clear all;
close all;
global C M0 F
```

```
ts = 0.02;
T = 30;
TimeSet = [0:ts:T];

para = [];
options = odeset('RelTol',1e-3,'AbsTol',[1e-3 1e-3 1e-3 1e-3]);
% options = [];
x0 = [0.5,0.3,0,0];
[t,xout] = ode45('chap2_3eq',TimeSet,x0,options,para);
x1 = xout(:,1);
x2 = xout(:,2);
x3 = xout(:,3);
x4 = xout(:,4);

s = C(1) * x1 + C(2) * x2 + C(3) * x3 + C(4) * x4;

if F == 1
   M0 = 40;
   u = -M0 * sign(s);
elseif F == 2
   beta = 30;
   delta = 0;
   for k = 1:1:T/ts + 1
u(k) = -beta * (abs(x1(k)) + abs(x2(k)) + abs(x3(k)) + abs(x4(k)) + delta) * sign(s(k));
   end
end

figure(1);
plot(t,x1,'r');
xlabel('time(s)');ylabel('Cart Position');
figure(2);
plot(t,x2,'r');
xlabel('time(s)');ylabel('Pendulum Angle');
figure(3);
plot(t,s,'r');
xlabel('time(s)');ylabel('s');
figure(4);
plot(t,u,'r');
xlabel('time(s)');ylabel('u');
```

(2) 控制子程序：chap2_3eq.m

```
function dx = DxnamicModel(t,x,flag,para)
global C M0 F

M = 5;m = 1;a = 1;C1 = 1;g = 9.81;
a32 = -3 * (C1 - m * g * a)/(a * (4 * M + m));
a42 = -3 * (M + m) * (C1 - m * g * a)/(a^2 * m * (4 * M + m));
b3 = 4/(4 * M + m);
b4 = 3/(4 * M + m);
```

```
A = [0,0,1,0;
   0,0,0,1;
   0,a32,0,0;
   0,a42,0,0];
b = [0;0;b3;b4];

%Ackermann's formula
n1 = -1;n2 = -2;n3 = -3;
C = [0,0,0,1] * inv([b,A * b,A^2 * b,A^3 * b]) * (A - n1 * eye(4)) * (A - n2 * eye(4)) * (A - n3 * eye
(4));
s = C * x;

F = 2;
if F == 1
   M0 = 40;
   u = - M0 * sign(s);
elseif F == 2
   beta = 30;
   delta = 0;
   u = - beta * (abs(x(1)) + abs(x(2)) + abs(x(3)) + abs(x(4)) + delta) * sign(s);
end

%State equation
dx = zeros(4,1);
f0 = 0.5;
ft = f0 * sin(3 * t);

dx = A * x + b * (u + ft);
```

2.7　用趋近律方法设计滑模控制器

滑模运动包括趋近运动和滑模运动两个过程。系统从任意初始状态趋向切换面,直到到达切换面的运动称为趋近运动,即趋近运动为 $s \to 0$ 的过程。根据滑模变结构原理,滑模可达性条件仅保证由状态空间任意位置运动点在有限时间内到达切换面的要求,而对于趋近运动的具体轨迹未作任何限制,采用趋近律的方法可以改善趋近运动的动态品质。

2.7.1　几种典型的趋近律

(1) 等速趋近律

$$\dot{s} = -\varepsilon \operatorname{sgn}(s) \quad \varepsilon > 0 \tag{2.37}$$

其中,常数 ε 表示系统的运动点趋近切换面 $s=0$ 的速率。ε 小,趋近速度慢;ε 大,则运动点到达切换面时将具有较大的速度,引起的抖振也较大。

(2) 指数趋近律

$$\dot{s} = -\varepsilon \operatorname{sgn}(s) - ks \qquad \varepsilon > 0, k > 0 \tag{2.38}$$

式中，$\dot{s}=-ks$ 是指数趋近项，其解为 $s=s(0)e^{-kt}$。

指数趋近中，趋近速度从一较大值逐步减小到零，不仅缩短了趋近时间，而且使运动点到达切换面时的速度很小。对单纯的指数趋近，运动点逼近切换面是一个渐近的过程，不能保证有限时间内到达，切换面上也就不存在滑动模态了，所以要增加一个等速趋近项 $\dot{s}=-\varepsilon\mathrm{sgn}(s)$，使当 s 接近于零时，趋近速度是 ε 而不是零，可以保证有限时间到达。

在指数趋近律中，为了保证快速趋近的同时削弱抖振，应在增大 k 的同时减小 ε。

（3）幂次趋近律

$$\dot{s}=-k|s|^{\alpha}\mathrm{sgn}(s) \qquad k>0,\ 1>\alpha>0 \tag{2.39}$$

（4）一般趋近律

$$\dot{s}=-\varepsilon\mathrm{sgn}(s)-f(s) \quad \varepsilon>0 \tag{2.40}$$

其中 $f(0)=0$，当 $s\neq0$ 时，$sf(s)>0$。

显然，上述四种趋近律都满足滑模到达条件 $s\dot{s}<0$。

2.7.2 基于趋近律的滑模控制

1. 控制器的设计

针对状态方程

$$\dot{\boldsymbol{x}}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}u \tag{2.41}$$

采用趋近律的控制方式，控制律推导如下：

$$s=\boldsymbol{C}\boldsymbol{x}$$
$$\dot{s}=\boldsymbol{C}\dot{\boldsymbol{x}}=\mathrm{slaw} \tag{2.42}$$

其中 slaw 为趋近律。

将状态方程式(2.41)代入式(2.42)得

$$u=(\boldsymbol{CB})^{-1}(-\boldsymbol{CAx}+\dot{s}) \tag{2.43}$$

可见，控制器的抖振程度取决于趋近律 $\dot{s}$ 表达式中的切换项。

2. 仿真实例

对象为二阶传递函数：

$$G_p(s)=\frac{b}{s^2+as}$$

其中 $a=25$，$b=133$。

$G_p(s)$ 可表示为如下状态方程：

$$\dot{\boldsymbol{x}}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}u$$

其中 $\boldsymbol{A}=\begin{bmatrix}0 & 1\\0 & -25\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0\\133\end{bmatrix}$。

在仿真程序中，$M=1$ 为等速趋近律，$M=2$ 为指数趋近律，$M=3$ 为幂次趋近律，$M=4$ 为一般趋近律。取 $M=2$，采用指数趋近律，其中 $\boldsymbol{C}=[15,1]$，$\varepsilon=5$，$k=10$，作图取样时间为 0.001，仿真结果如图 2-16～图 2-20 所示。

仿真程序如下。

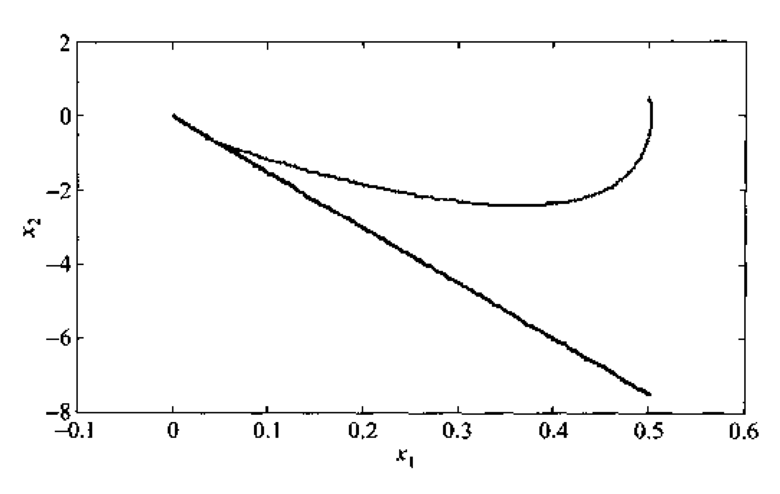


图 2-16　滑模运动的相轨迹

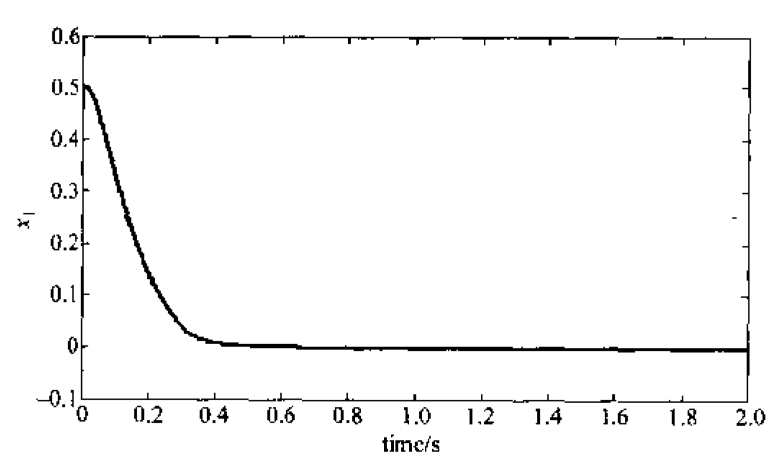


图 2-17　x_1 的收敛过程

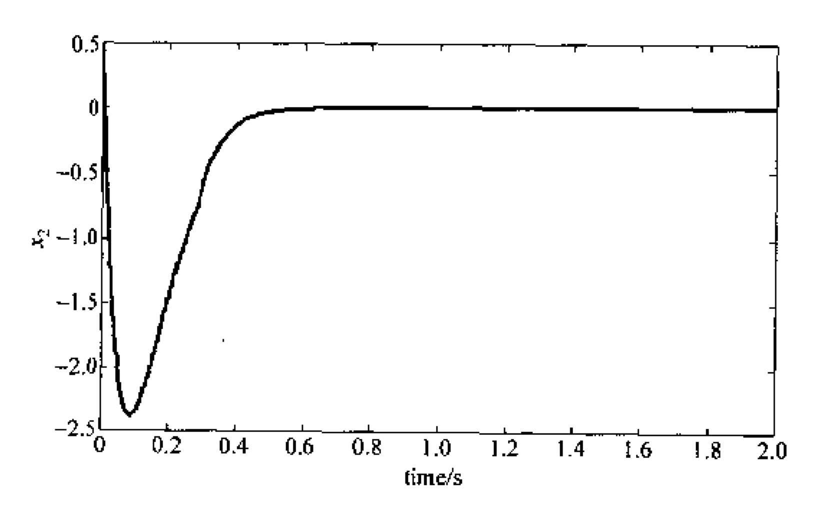


图 2-18　x_2 的收敛过程

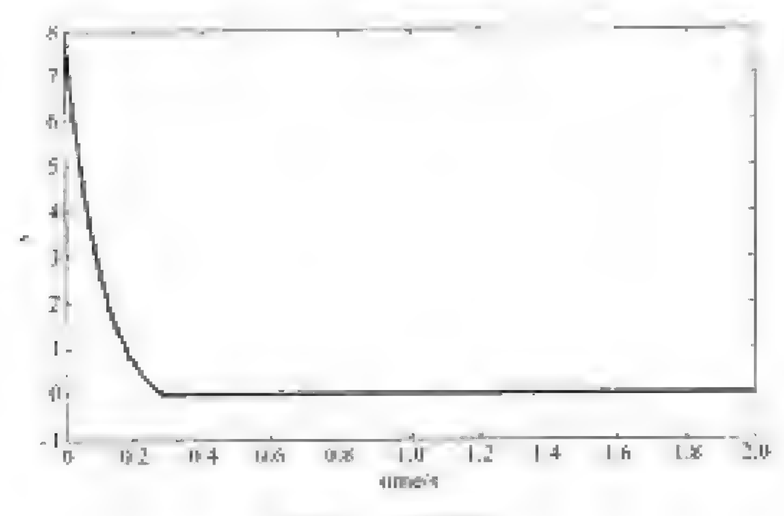


图 2-19 切换函数 s

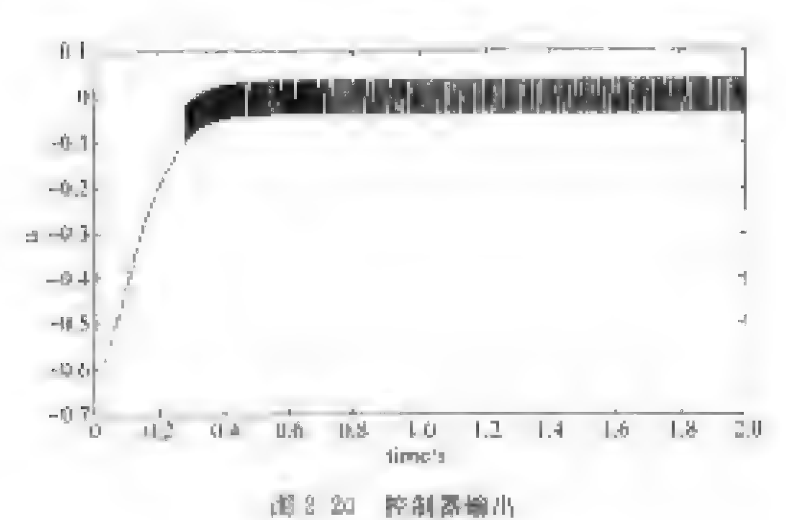


图 2-20 控制器输出

(1) 主程序：chap2_4.m

```
clear all;
close all;
global M A B C eq k
ts = 0.001;
T = 2;
TimeSet = [0:ts:T];

c = 15;
C = [c,1];
para = [];

[t,x] = ode45('chap2_4eq',TimeSet,[0.50 0.50],[],para);
x1 = x(:,1);
x2 = x(:,2);

s = c * x(:,1) + x(:,2);
```

```
if M == 2
    for kk = 1:1:T/ts + 1
        xk = [x1(kk);x2(kk)]';
        sk(kk) = c * x1(kk) + x2(kk);
        slaw(kk) = - eq * sign(sk(kk)) - k * sk(kk);        % Exponential trending law
        u(kk) = inv(C * B) * ( - C * A * xk + slaw(kk));
    end
end

figure(1);
plot(x(:,1),x(:,2),'r',x(:,1), - c * x(:,1),'b');
xlabel('x1');ylabel('x2');
figure(2);
plot(t,x(:,1),'r');
xlabel('time(s)');ylabel('x1');
figure(3);
plot(t,x(:,2),'r');
xlabel('time(s)');ylabel('x2');
figure(4);
plot(t,s,'r');
xlabel('time(s)');ylabel('s');
if M == 2
    figure(5);
    plot(t,u,'r');
    xlabel('time(s)');ylabel('u');
end
```

(2) 控制子程序：chap2_4eq. m

```
function dx = DynamicModel(t,x,flag,para)
global M A B C eq k
a = 25;b = 133;

c = para(1);
s = c * x(1) + x(2);

A = [0  1;0  - a];
B = [0;b];

M = 2;
eq = 5.0;
if M == 1
    slaw = - eq * sign(s);                   % Equal velocity trending law
elseif M == 2
    k = 10;
    slaw = - eq * sign(s) - k * s;           % Exponential velocity trending law
elseif M == 3
    k = 10;
    alfa = 0.50;
    slaw = - k * abs(s)^alfa * sign(s);      % Power trending law
elseif M == 4
```

```
    k = 1;
    slaw = - eq * sign(s) - k * s^3;          % General trending law
end

u = inv(C * B) * ( - C * A * x + slaw);

dx = zeros(2,1);
dx(1) = x(2);
dx(2) = - a * x(2) + b * u;
```

2.7.3 基于趋近律的位置跟踪

1. 控制器的设计

针对状态方程

$$\dot{x} = Ax + Bu \tag{2.44}$$

其中 $x=\begin{bmatrix} x(1) \\ x(2) \end{bmatrix}$，$A=\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$，$B=\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$。则

$$\dot{x}(2) = A_{21}x(1) + A_{22}x(2) + B_2 u$$

设指令信号为 r，则误差为

$$e = r - x(1)$$

误差变化率为

$$\dot{e} = \dot{r} - x(2)$$

设 $C=[c,1]$，误差向量为 $E=\begin{bmatrix} e \\ \dot{e} \end{bmatrix}$，则切换函数为

$$s = CE = c(r - x(1)) + \dot{r} - x(2)$$

$$\dot{s} = c(\dot{r} - \dot{x}(1)) + \ddot{r} - \dot{x}(2) = \text{slaw}$$

其中 slaw 为趋近律。

将状态方程代入 $\dot{s}$ 中，得控制律

$$u = \frac{1}{B_2}(c(\dot{r} - x(2)) + \ddot{r} - A_{21}x(1) - A_{22}x(2) - \text{slaw}) \tag{2.45}$$

2. 仿真实例

对象为二阶传递函数

$$G_p(s) = \frac{b}{s^2 + as}$$

其中 $a=25$，$b=133$。

可表示为如下状态方程的形式：

$$\dot{x} = Ax + Bu$$

其中 $A=\begin{bmatrix} 0 & 1 \\ 0 & -25 \end{bmatrix}$，$B=\begin{bmatrix} 0 \\ 133 \end{bmatrix}$。

在程序中，$S=1$ 时指令为正弦信号，$S=2$ 时指令为方波信号，$S=3$ 时指令为正弦叠加信号。采用指数趋近律。其中 $\boldsymbol{C}=[5,1]$，$\varepsilon=5.0$，$k=10$。

(1) 取 $S=1$，正弦跟踪结果如图 2-21～图 2-23 所示。

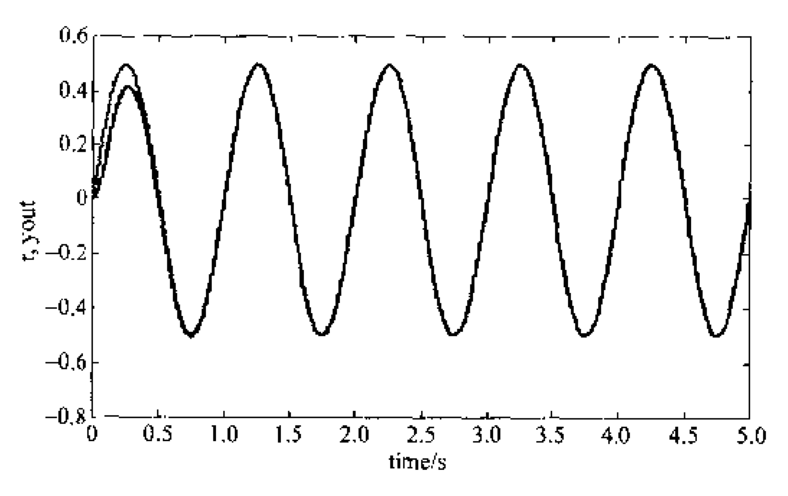


图 2-21　正弦跟踪（S=1）

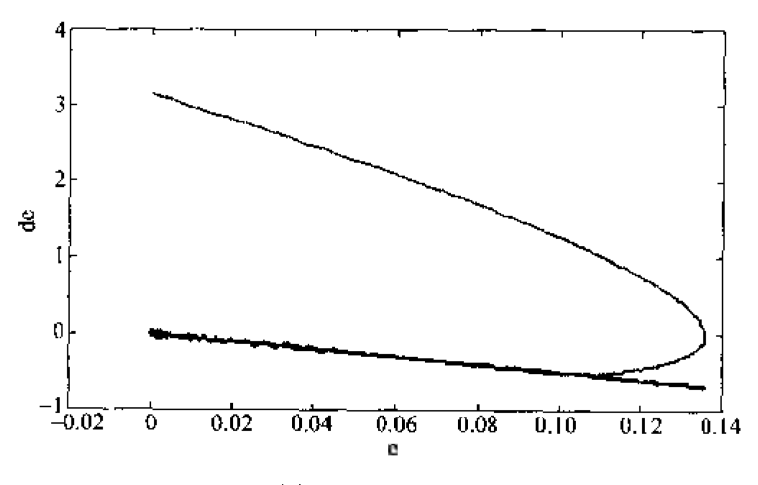


图 2-22　相轨迹

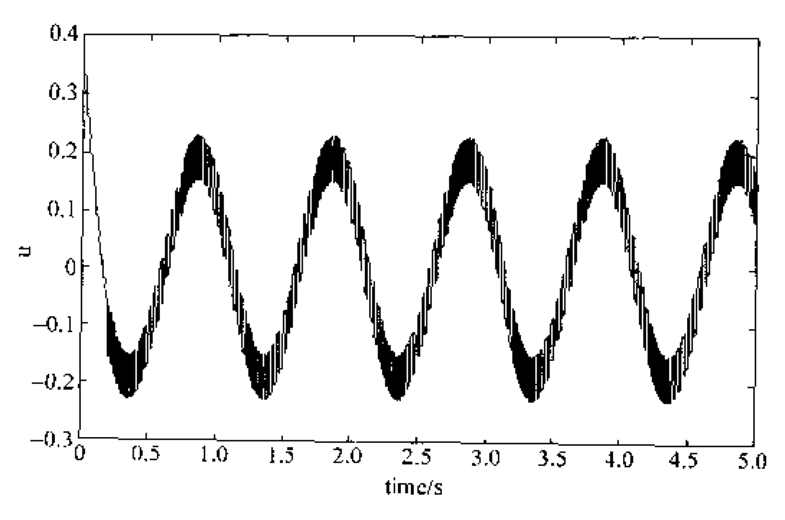


图 2-23　控制输入信号

(2) 取 $S=2,3$,方波和正弦叠加信号的位置跟踪结果如图 2-24 和图 2-25 所示。

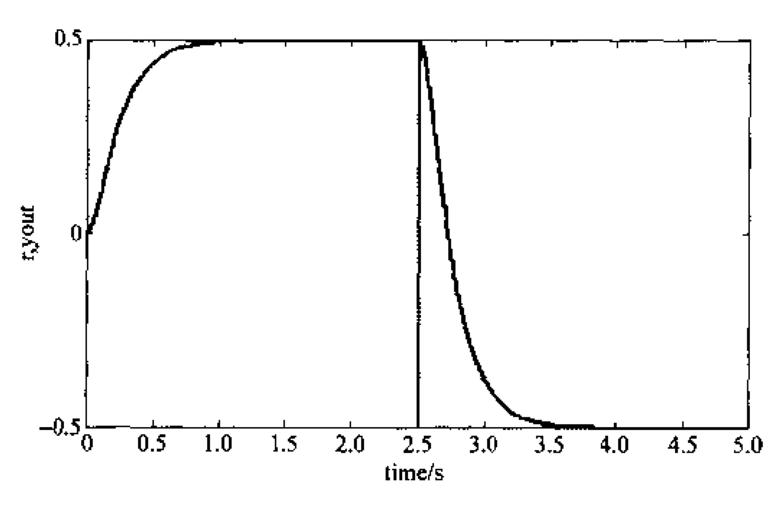


图 2-24 方波跟踪

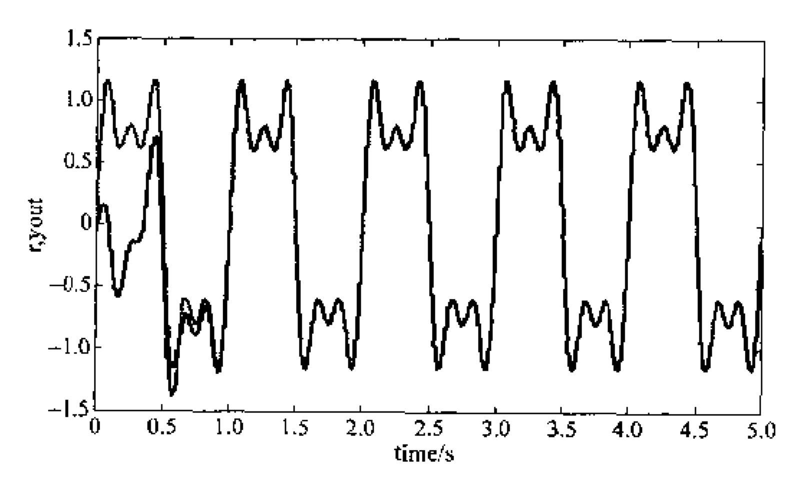


图 2-25 正弦叠加信号跟踪

仿真程序如下。

(1) 主程序:chap2_5.m

```
clear all;
close all;
global a b c S A F ep k delta
ts = 0.001;
T = 5;
TimeSet = [0:ts:T];

c = 5.0;
para = [];

[t,x] = ode45('chap2_5eq',TimeSet,[0 0],[],para);
x1 = x(:,1);
x2 = x(:,2);

if S == 1
```

```
    r = A * sin(2 * pi * F * t);
    dr = A * 2 * pi * F * cos(2 * pi * F * t);
     ddr = - A * (2 * pi * F)^2 * sin(2 * pi * F * t);
elseif S == 2
     r = A * sign(sin(2 * pi * F * t));
     dr = 0;ddr = 0;
elseif S == 3
    r = 1.0 * sin(2 * pi * 1 * t) + 0.5 * sin(2 * pi * 3 * t) + 0.3 * sin(2 * pi * 5 * t);

    dr = 1.0 * 2 * pi * 1 * cos(2 * pi * 1 * t) + 0.50 * 2 * pi * 3 * cos(2 * pi * 3 * t) + 0.30 * 2 * pi *
5 * cos(2 * pi * 5 * t);

    ddr = - 1.0 * (2 * pi * 1)^2 * sin(2 * pi * 1 * t) - 0.50 * (2 * pi * 3)^2 * sin(2 * pi * 3 * t)
 - 0.30 * (2 * pi * 5)^2 * sin(2 * pi * 5 * t);
end
s = c * (r - x(:,1)) + dr - x(:,2);

slaw = - ep * sign(s) - k * s;        % Exponential velocity trending law
u = 1/b * (c * (dr - x(2)) + ddr - slaw + a * x(2));

figure(1);
plot(t,r,'r',t,x(:,1),'b');
xlabel('time(s)');ylabel('r,yout');
figure(2);
plot(t,r - x(:,1),'r');
xlabel('time(s)');ylabel('error');
figure(3);
plot(r - x(:,1),dr - x(:,2),'r',r - x(:,1), - c * (r - x(:,1)),'b');
xlabel('e');ylabel('de');
figure(4);
plot(t,s,'r');
xlabel('time(s)');ylabel('s');
figure(5);
plot(t,u,'r');
xlabel('time(s)');ylabel('u');
```

(2) 控制子程序：chap2_5eq.m

```
function dx = DynamicModel(t,x,flag,para)
global a b c S A F ep k delta

a = 25;b = 133;

S = 3;
if S == 1            % Sin Signal
   A = 0.50;F = 1.0;
   r = A * sin(2 * pi * F * t);
    dr = A * 2 * pi * F * cos(2 * pi * F * t);
    ddr = - A * (2 * pi * F)^2 * sin(2 * pi * F * t);
elseif S == 2       % Square Signal
   A = 0.50;F = 0.20;
    r = A * sign(sin(2 * pi * F * t));
```

```
    dr = 0;ddr = 0;
elseif S == 3
   A = 0.50;F = 1.0;
   r = 1.0 * sin(2 * pi * 1 * t) + 0.5 * sin(2 * pi * 3 * t) + 0.3 * sin(2 * pi * 5 * t);

dr = 1.0 * 2 * pi * 1 * cos(2 * pi * 1 * t) + 0.50 * 2 * pi * 3 * cos(2 * pi * 3 * t) + 0.30 * 2 * pi * 5 *
cos(2 * pi * 5 * t);

ddr = - 1.0 * (2 * pi * 1)^2 * sin(2 * pi * 1 * t) - 0.50 * (2 * pi * 3)^2 * sin(2 * pi * 3 * t) - 0.30 *
(2 * pi * 5)^2 * sin(2 * pi * 5 * t);
end

s = c * (r - x(1)) + dr - x(2);

k = 10;
ep = 5;

slaw = - ep * sign(s) - k * s;        % Exponential trending law
u = 1/b * (c * (dr - x(2)) + ddr - slaw + a * x(2));

% State Equation
dx = zeros(2,1);
dx(1) = x(2);
dx(2) = - a * x(2) + b * u;
```

2.8 准滑动模态控制

2.8.1 准滑动模态控制原理

在滑动模态控制系统中，如果控制结构的切换具有理想的开关特性，则能在切换面上形成理想的滑动模态，这是一种光滑的运动，渐进趋近于原点。但在实际工程中，由于存在时间上的延迟和空间上的滞后等原因，使得滑动模态呈抖振形式，在光滑的滑动上叠加了抖振。理想的滑动模态是不存在的，现实中的滑动模态控制均伴随有抖振，抖振问题是影响滑动模态控制广泛应用的主要障碍。

所谓准滑动模态，是指系统的运动轨迹被限制在理想滑动模态的某一 Δ 邻域内的模态。从相轨迹方面来说，具有理想滑动模态的控制是使一定范围内的状态点均被吸引至切换面。而准滑动模态控制则是使一定范围内的状态点均被吸引至切换面的某一 Δ 邻域内，通常称此 Δ 邻域为滑动模态切换面的边界层。

在边界层内，准滑动模态不要求满足滑动模态的存在条件，因此准滑动模态不要求在切换面上进行控制结构的切换。它可以是在边界层上进行结构变换的控制系统，也可以是根本不进行结构变换的连续状态反馈控制系统。准滑动模态控制在实现上的这种差别，使它从根本上避免或削弱了抖振，从而在实际中得到了广泛的应用。

在连续系统中，常用的准滑动模态控制有以下两种方法：

（1）用饱和函数 $\mathrm{sat}(s)$ 代替理想滑动模态中的符号函数 $\mathrm{sgn}(s)$。

$$\mathrm{sat}(s)=\begin{cases}1 & s>\Delta\\ ks & |s|\leqslant\Delta \quad k=\dfrac{1}{\Delta}\\ -1 & s<-\Delta\end{cases} \tag{2.46}$$

其中 Δ 称为“边界层”。饱和函数 sat(s)如图 2-26 所示。饱和函数的本质为：在边界层外，采用切换控制；在边界层之内，采用线性化反馈控制。

(2) 将继电特性连续化，用连续函数 $\theta(s)$取代 sgn(s)。

$$\theta(s)=\frac{s}{|s|+\delta} \tag{2.47}$$

式中，δ 是很小的正常数。

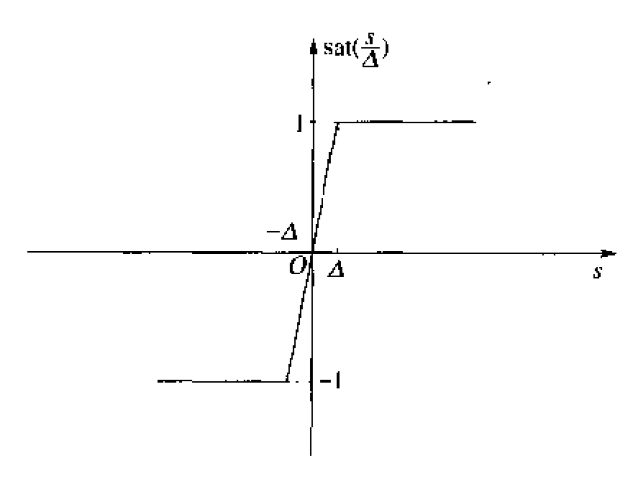


图 2-26 饱和函数

2.8.2 仿真实例

对象为二阶传递函数：

$$G_p(s)=\frac{b}{s^2+as}$$

其中 $a=25$，$b=133$。

将传递函数描述为位置状态方程的形式：

$$\dot{\boldsymbol{x}}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}u$$

其中 $\boldsymbol{A}=\begin{bmatrix}0 & 1\\ 0 & -a\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0\\ b\end{bmatrix}$。

设位置指令信号为 r，则

$$s=c[r-\boldsymbol{x}(1)]+\mathrm{d}r-\boldsymbol{x}(2)$$
$$\dot{s}=c[\dot{r}-\dot{\boldsymbol{x}}(1)]+\ddot{r}-\dot{\boldsymbol{x}}(2)=\mathrm{slaw}$$

采用指数趋近律：

$$\mathrm{slaw}=-\varepsilon\,\mathrm{sgn}(s)-ks$$

将 $\dot{\boldsymbol{x}}(2)=-a\,\boldsymbol{x}(2)+bu$ 代入上式，得控制律为

$$u=\frac{1}{b}[c(\dot{r}-\boldsymbol{x}(2))+\ddot{r}-\mathrm{slaw}+a\boldsymbol{x}(2)]$$

系统的初始状态为[−0.5,0]。在仿真程序中，指令为正弦信号，$M=1$ 为指数趋近律，$M=2$ 为采用饱和函数的指数趋近律，$M=3$ 为采用继电特性的指数趋近律。取 $C=5$，$\varepsilon=15$，$k=30$，进行如下几种仿真。

(1) 采用指数趋近律，取 $M=1$，正弦跟踪结果如图 2-27～图 2-29 所示。

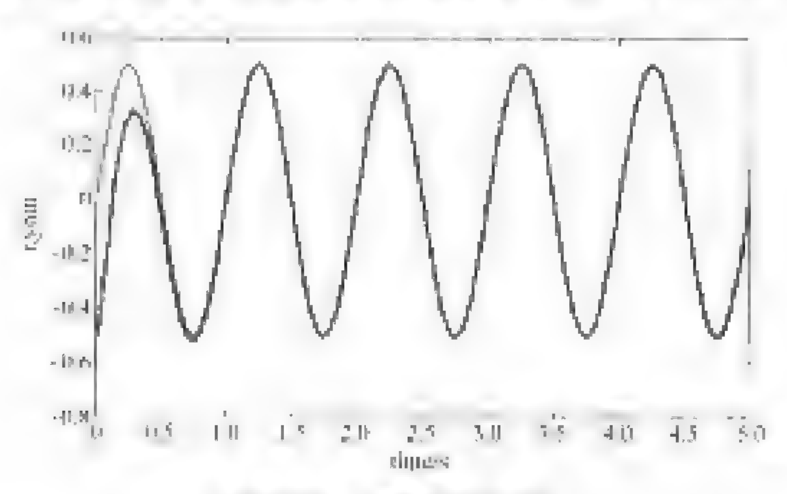


图 2-27 正弦跟踪($M=1$)

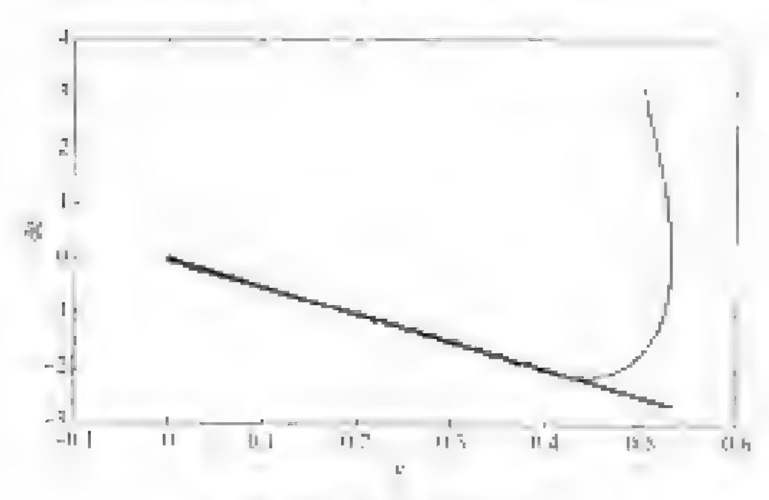


图 2-28 相轨迹($M=1$)

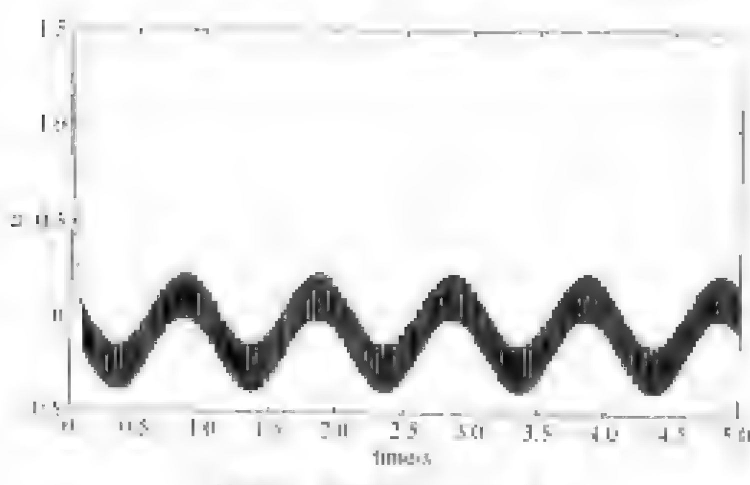


图 2-29 控制器输出($M=1$)

（2）验证准滑动模态控制。$M=2$ 为采用饱和函数的方法，取 $\Delta=0.05$，仿真结果如图 2-30 及图 2-31 所示。$M=3$ 为采用继电特性的方法，取 $\delta=0.05$，仿真结果如图 2-32 及图 2-33 所示。

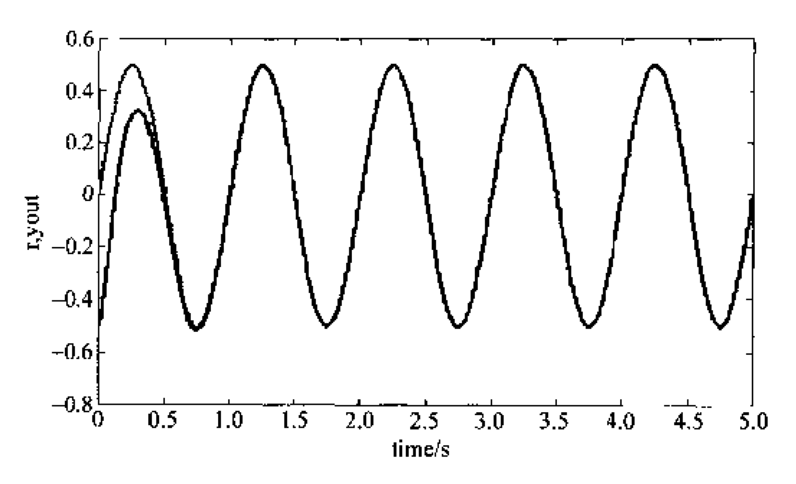


图 2-30　正弦信号跟踪（$M=2$）

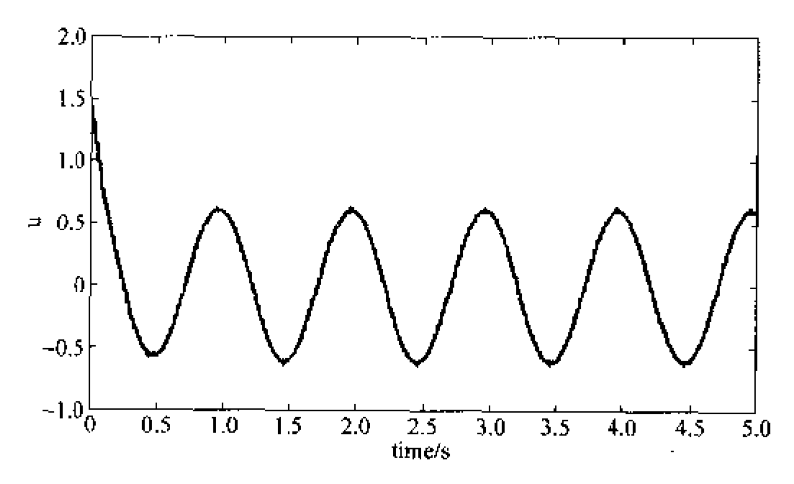


图 2-31　控制器输出（$M=2$）

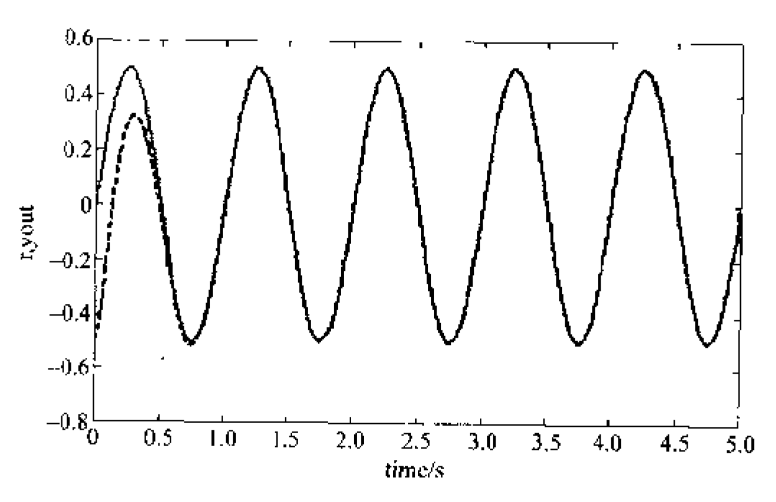


图 2-32　正弦信号跟踪（$M=3$）

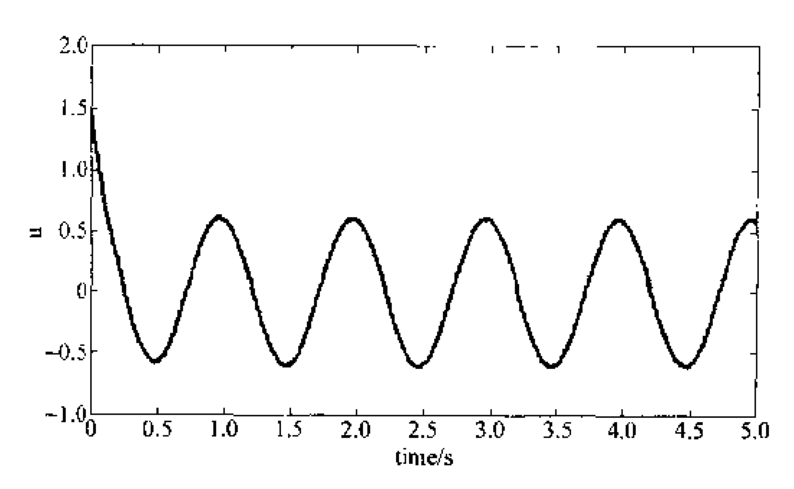


图 2-33 控制器输出($M=3$)

仿真程序如下。

(1) 主程序：chap2_6. m

```
clear all;
close all;
global a b c A F M ep k delta
ts = 0.001;
T = 5;
TimeSet = [0:ts:T];

c = 5.0;
para = [];

[t,x] = ode45('chap2_6eq',TimeSet,[-0.5, 0],[],para);
x1 = x(:,1);
x2 = x(:,2);

r = A * sin(2 * pi * F * t);
dr = A * 2 * pi * F * cos(2 * pi * F * t);
ddr = -A * (2 * pi * F)^2 * sin(2 * pi * F * t);

s = c * (r - x(:,1)) + dr - x(:,2);

if M == 1
    slaw = - ep * sign(s) - k * s;          % Exponential velocity trending law
    u = 1/b * (c * (dr - x(2)) + ddr - slaw + a * x(2));
elseif M == 2                              % Saturated function
    kk = 1/delta;
    for i = 1:1:T/ts + 1
   if s(i)>delta
        sats(i) = 1;
   elseif abs(s(i))<= delta
        sats(i) = kk * s(i);
   elseif s(i)<-delta
        sats(i) = -1;
```

```
    end
    slaw(i) = - ep * sats(i) - k * s(i);
        u(i) = 1/b * (c * (dr(i) - x2(i)) + ddr(i) - slaw(i) + a * x2(i));
    end

elseif M == 3                              % Relay characteristics
    for i = 1:1:T/ts + 1
    ths(i) = s(i)/(abs(s(i)) + delta);
        slaw(i) = - ep * ths(i) - k * s(i);
        u(i) = 1/b * (c * (dr(i) - x2(i)) + ddr(i) - slaw(i) + a * x2(i));
    end
end

figure(1);
plot(t,r,'r',t,x(:,1),'b');
xlabel('time(s)');ylabel('r,yout');
figure(2);
plot(t,r - x(:,1),'r');
xlabel('time(s)');ylabel('error');
figure(3);
plot(r - x(:,1),dr - x(:,2),'r',r - x(:,1), - c * (r - x(:,1)),'b');     % Draw line(s = 0)
xlabel('e');ylabel('de');
figure(4);
plot(t,s,'r');
xlabel('time(s)');ylabel('s');
figure(5);
plot(t,u,'r');
xlabel('time(s)');ylabel('u');
```

(2) 控制子程序：chap2_6eq.m

```
function dx = DynamicModel(t,x,flag,para)
global a b c A F M ep k delta

a = 25;b = 133;

A = 0.50;F = 1.0;
r = A * sin(2 * pi * F * t);
dr = A * 2 * pi * F * cos(2 * pi * F * t);
ddr = - A * (2 * pi * F)^2 * sin(2 * pi * F * t);

s = c * (r - x(1)) + dr - x(2);

k = 30;
ep = 15;

M = 3;
if M == 1
    slaw = - ep * sign(s) - k * s;       % Exponential trending law
elseif M == 2                            % Saturated function
    delta = 0.05;
```

```
    kk = 1/delta;
    if s>delta
        sats = 1;
    elseif abs(s)<= delta
        sats = kk * s;
    elseif s< - delta
        sats = - 1;
    end
    slaw = - ep * sats - k * s;
elseif M == 3                              % Relay characteristics
    delta = 0.05;
    ths = s/(abs(s) + delta);
    slaw = - ep * ths - k * s;
end

u = 1/b * (c * (dr - x(2)) + ddr - slaw + a * x(2));

% State Equation
dx = zeros(2,1);
dx(1) = x(2);
dx(2) = - a * x(2) + b * u;
```

2.9 滑模控制在低速摩擦伺服系统中的应用

2.9.1 伺服系统摩擦模型

摩擦存在于所有的运动中，特别是对高性能伺服系统的影响尤为突出。对于伺服系统来说，摩擦是影响系统低速性能的重要因素，它不但造成系统的稳态误差，而且使系统产生爬行、振荡。从控制的角度来说，为了减轻机械伺服系统中摩擦环节引起的不良影响，可采用滑模变结构控制方法[2]。

在机械伺服系统中，具有相对运动或相对运动趋势的两个接触面上会产生摩擦力，接触面由相对静止到相对运动经过四个阶段，分别为接触面弹性形变阶段、边界润滑阶段、部分液体润滑阶段及完全液体润滑阶段。在不同的阶段中，接触面之间的相对运动速度是不同的，因此在稳态时，摩擦力表现为相对速度的函数，通常称这种稳态对应关系为 Stribeck 曲线。在第一阶段，接触面之间没有相对滑动，只是发生轻微的弹性形变，此时的摩擦力称为静摩擦力，摩擦力与所施加的控制力大小相同，方向相反。当所施加的外力达到某一临界值后，开始第二阶段边界润滑阶段。从这一阶段到完全液体润滑阶段，接触面间由直接接触直至接触面完全脱离。部分液体润滑阶段正体现了这一过渡过程，此时，摩擦力随着速度的增加而减小，即对应 Stribeck 曲线中的斜率为负的部分。当完全润滑阶段开始后，随着速度的增加，与速度成正比的粘性摩擦力(矩)渐渐占据主导作用。

对于机械角位置伺服系统，摩擦环节会产生如下不良的影响[4]：

(1) 零速时存在的静摩擦将使系统响应表现出死区特性，系统的稳态响应具有多个平

衡点，为一条线段，使系统存在很大的静态误差。

(2) 当静摩擦大于库仑摩擦时，引入的积分控制不能消除静差，而且会使系统响应出现极限振荡现象。这是由于含有摩擦环节的伺服系统从静止到运动的过程中，摩擦的变化是不连续的且具有负斜率特性。

(3) 在低速情况下，当位置输入为斜坡信号时，系统会出现静、动、静、动……的跳跃运动，即低速爬行的现象。

(4) 在零速的时候，由于静摩擦是不连续的，并且多值，使系统在速度过零时的运动不平稳，出现“平顶”现象。

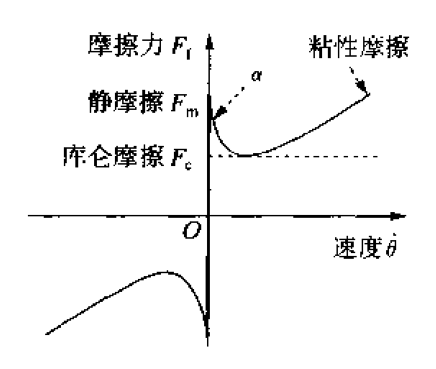


图 2-34 摩擦-速度稳态关系曲线（Stribeck 曲线）

Stribeck 曲线是比较著名的摩擦模型。如图 2-34 所示，该图表明了在不同的摩擦阶段，摩擦力矩与速度之间的关系，该关系即为 Stribeck 曲线。

Stribeck 摩擦模型可表示如下：

当 $|\dot{\theta}(t)|<\alpha$ 时，静摩擦为

$$F_f(t)=\begin{cases}F_m & F(t)>F_m\\ F(t) & -F_m<F<F_m\\ -F_m & F(t)<-F_m\end{cases}\tag{2.48}$$

当 $|\dot{\theta}(t)|>\alpha$ 时，动摩擦为

$$F_f(t)=[F_c+(F_m-F_c)\mathrm{e}^{-\alpha_1|\dot{\theta}(t)|}]\mathrm{sgn}(\dot{\theta}(t))+k_v\dot{\theta}\tag{2.49}$$

$$F(t)=J\ddot{\theta}(t)\tag{2.50}$$

式中，$F(t)$ 为驱动力，F_m 为最大静摩擦力，F_c 为库仑摩擦力，k_v 为粘性摩擦力矩比例系数，$\dot{\theta}(t)$ 为转动角速度，α 和 α_1 为非常小的、正的常数。

2.9.2 一个典型伺服系统描述

以飞行模拟转台伺服系统为例，它是三轴伺服系统，其任意框的模型在正常情况下可简化为线性二阶环节的系统，在低速情况下具有较强的摩擦现象，此时控制对象就变为非线性，很难用传统控制方法达到高精度控制[4,5]。任意框的伺服结构如图 2-35 所示，该系统采用直流电机，忽略电枢电感，电流环和速度环为开环。其中 K_u 为 PWM 功率放大器放大系数，R 为电枢电阻，K_m 为电机力矩系数，C_e 为电压反馈系数，J 为该框的转动惯量，$\dot{\theta}(t)$ 为转速，$r(t)$ 为指令信号，$u(t)$ 为控制输入。

根据伺服系统的结构，飞行模拟转台位置状态方程可描述如下：

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix}=\begin{bmatrix}0 & 1\\ 0 & -\dfrac{K_mC_e}{JR}\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}+\begin{bmatrix}0\\ K_u\dfrac{K_m}{JR}\end{bmatrix}u(t)-\begin{bmatrix}0\\ \dfrac{1}{J}\end{bmatrix}F_f(t)\tag{2.51}$$

式中，$x_1(t)=\theta(t)$ 为转角，$x_2(t)=\dot{\theta}(t)$ 为转速。

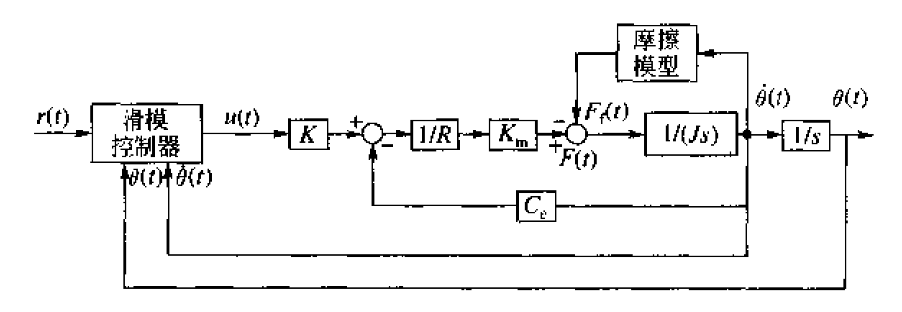


图 2-35　转台伺服系统结构

2.9.3　滑模控制器设计

设计切换函数

$$s = ce + \dot{e} \tag{2.52}$$

采用指数趋近律

$$\dot{s} = -\varepsilon \operatorname{sgn}(s) - ks \tag{2.53}$$

其中 $\varepsilon>0$，$k>0$。考虑系统的摩擦力 F_f 的影响，由式(2.51)得

$$\begin{aligned}\dot{s} &= c\dot{e} + \ddot{e} = c\dot{e} + \ddot{r} - \ddot{x} \\ &= c\dot{e} + \ddot{r} - \left(-\frac{K_m C_e}{JR}\dot{x} + K_u \frac{K_m}{JR}u - \frac{F_f}{J}\right)\end{aligned} \tag{2.54}$$

由式(2.53)和式(2.54)得

$$u = \frac{JR}{K_u K_m}\left(c\dot{e} + \ddot{r} + \varepsilon \operatorname{sgn}(s) + ks + \frac{K_m C_e}{JR}\dot{x} + \frac{F_f}{J}\right) \tag{2.55}$$

2.9.4　仿真实例

设某转台某框伺服系统参数如下：

$R=7.77\Omega$，$K_m=6\text{Nm/A}$，$C_e=1.2\text{V/(rad/s)}$，$J=0.6\text{kgm}^2$，

$K_u=11$，$F_c=15\text{Nm}$，$F_m=20\text{N}\cdot\text{m}$，$k_v=2.0\text{Nms/rad}$，$a_1=1.0$，$\alpha=0.01$。

指令为正弦信号 $r(t)=0.10\sin(2\pi t)$。

采用 PD 控制：

$$u(t) = 100e(t) + 10\dot{e}(t) \tag{2.56}$$

采用 Simulink 进行仿真，采用 S 函数描述被控对象和控制器。分别采用 PD 控制和滑模控制。$M=1$ 时为 PD 控制，仿真结果如图 2-36 和图 2-37 所示。仿真结果表明在带有摩擦条件下，位置跟踪存在“平顶”现象，速度跟踪存在“死区”现象。采用 PID 控制鲁棒性差，不能达到高精度跟踪。$M=2$ 时为滑模控制，控制参数取 $c=30$，$k=5$，$\varepsilon=10$，仿真结果如图 2-38～图 2-42 所示。可见，采用滑模控制可消除摩擦造成的“平顶”和“死区”现象。

通过对两种仿真方法进行比较可知，采用基于 S 函数的 Simulink 仿真具有编程简单、运行速度快、对中间变量作图方便的特点。

仿真程序如下。

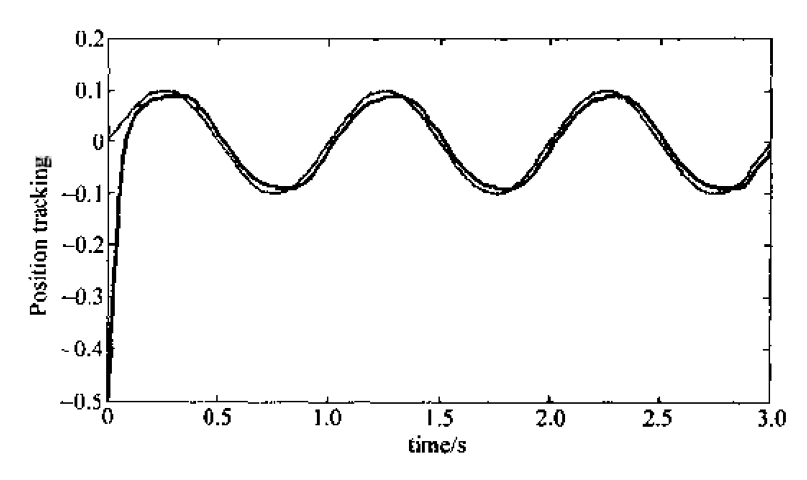


图 2-36　PD 位置跟踪（$M=1$）

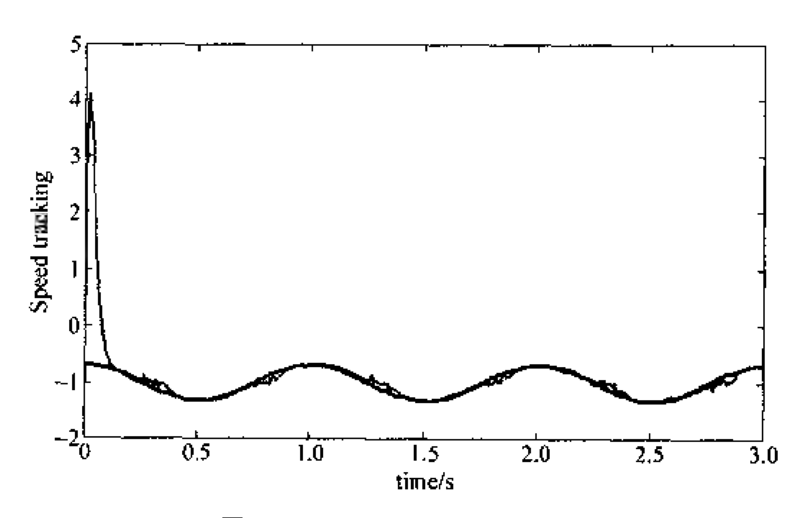


图 2-37　PD 速度跟踪（$M=1$）

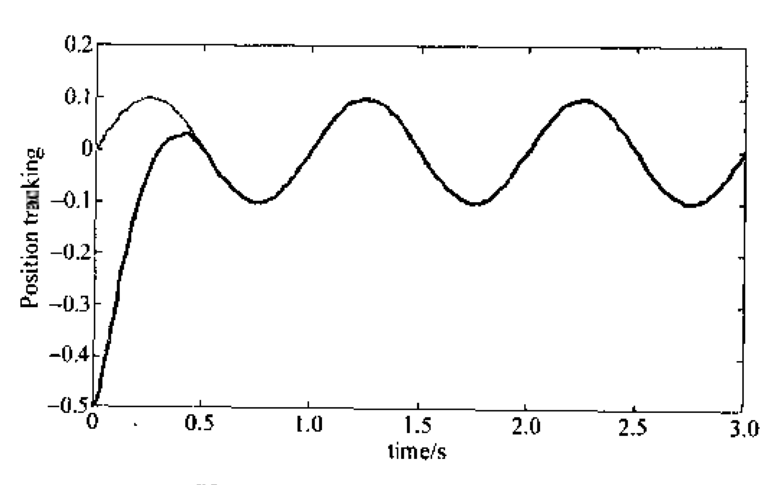


图 2-38　SMC 位置跟踪（$M=2$）

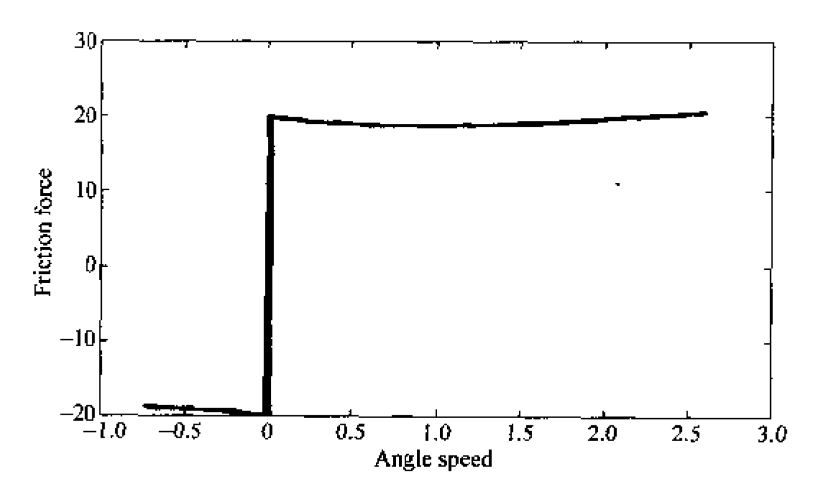


图 2-39 摩擦力变化曲线（$M=2$）

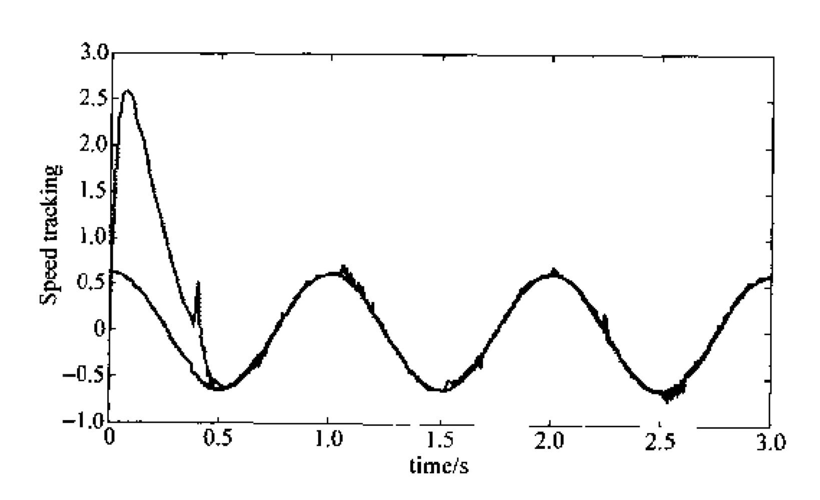


图 2-40 SMC 速度跟踪（$M=2$）

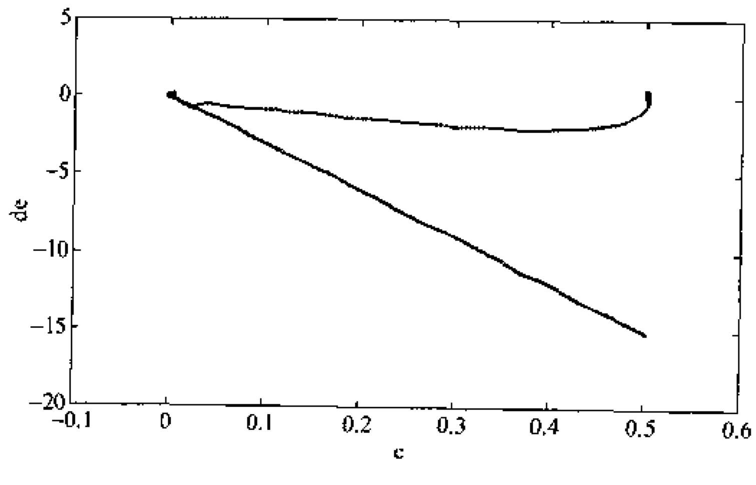


图 2-41 相轨迹（$M=2$）

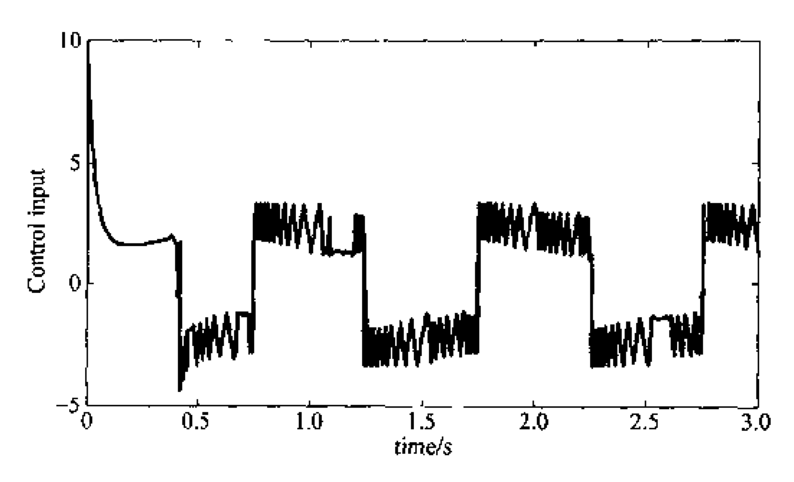


图 2-42　控制器输入信号（$M=2$）

（1）Simulink 主程序（如图 2-43 所示）：chap2_7.mdl

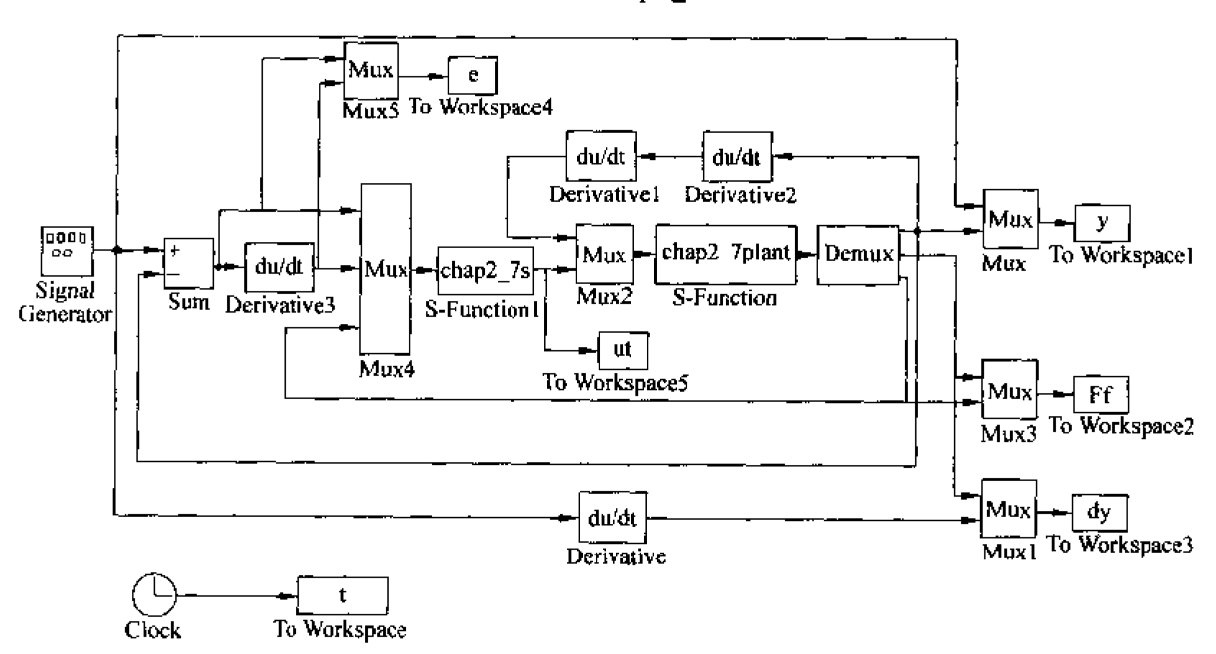


图 2-43　主程序图

（2）控制器 S 函数：chap2_7s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
```

```
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0 0];

function sys = mdlOutputs(t,x,u)
J = 0.6;Ce = 1.2;Km = 6;
Ku = 11;R = 7.77;

r = 0.1 * sin(2 * pi * t);
dr = 0.1 * 2 * pi * cos(2 * pi * t);
ddr = - 0.1 * (2 * pi)^2 * sin(2 * pi * t);

e = u(1);
de = u(2);
Ff = u(3);
x2 = dr - de;

c = 30;
eq = 10;
k = 5.0;
s = de + c * e;

M = 2;
if M == 1          % PD
   ut = 100 * e + 10 * de;
elseif M == 2      % SMC
   ut = J * R/(Ku * Km) * (c * de + ddr + eq * sign(s) + k * s + Km * Ce/(J * R) * x2 + Ff/J);
end

sys(1) = ut;
```

(3) 被控对象 S 函数：chap2_7plant.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
```

```
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1; % At least one sample time is needed
sys = simsizes(sizes);
x0  = [-0.5;0];
str = [];
ts  = [0 0];

function sys = mdlDerivatives(t,x,u)

%Servo system Parameters
J = 0.6;Ce = 1.2;Km = 6;
Ku = 11;R = 7.77;
kv = 2.0;

alfa = 0.01;
a1 = 1.0;   %Effect on the shape of friction curve
Fm = 20;
Fc = 15;

F = J * u(1);
if abs(x(2))<= alfa
   if F>Fm
      Ff = Fm;
   elseif F<-Fm
      Ff = -Fm;
   else
      Ff = F;
   end
end
if x(2)>alfa
   Ff = Fc + (Fm - Fc) * exp(-a1 * x(2)) + kv * x(2);
elseif x(2)<-alfa
   Ff = -Fc - (Fm - Fc) * exp(a1 * x(2)) + kv * x(2);
```

```
end

sys(1) = x(2);
sys(2) = -Km * Ce/(J * R) * x(2) + Ku * Km * u(2)/(J * R) - Ff/J;

function sys = mdlOutputs(t,x,u)

% Servo system Parameters
J = 0.6;Ce = 1.2;Km = 6;
Ku = 11;R = 7.77;
kv = 2.0;

alfa = 0.01;
a1 = 1.0;   % Effect on the shape of friction curve
Fm = 20;
Fc = 15;

F = J * u(1);
if abs(x(2))<= alfa
   if F>Fm
      Ff = Fm;
   elseif F< -Fm
      Ff = -Fm;
   else
      Ff = F;
   end
end
if x(2)>alfa
   Ff = Fc + (Fm - Fc) * exp( - a1 * x(2)) + kv * x(2);
elseif x(2)< - alfa
   Ff = - Fc - (Fm - Fc) * exp(a1 * x(2)) + kv * x(2);
end

sys(1) = x(1);     % Angle
sys(2) = x(2);     % Angle speed
sys(3) = Ff;       % Friction force
```

(4) 作图程序：chap2_7plot. m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(Ff(:,1),Ff(:,2),'r');
xlabel('Angle speed');ylabel('Friction force');
```

```
figure(3);
plot(t,dy(:,1),'r',t,dy(:,2),'b');
xlabel('time(s)');ylabel('Speed tracking');

figure(4);
c = 30;
plot(e(:,1),e(:,2),'r',e(:,1),-c*e(:,1),'b');
xlabel('e');ylabel('de');

figure(5);
plot(t,ut,'r');
xlabel('time(s)');ylabel('Control input');
```

2.10　一种基于上下界的滑模控制器设计

2.10.1　系统描述

系统的误差状态方程为

$$\left.\begin{aligned}\dot{x}_1(t) &= x_2(t)\\ \dot{x}_2(t) &= f(\boldsymbol{x},t)+\Delta f(\boldsymbol{x},t)+au(t)+d(t)\end{aligned}\right\} \tag{2.57}$$

其中 $\boldsymbol{x}(t)=[x_1(t),x_2(t)]$，$a$ 为常数，$d(t)$为外加干扰。

该系统具有以下特性：

(1) $d(t)$为系统的外部干扰，D 为 $d(t)$的上界，满足$|d(t)|\leqslant D$；

(2) $F(\boldsymbol{x},t)$为 $\Delta f(\boldsymbol{x},t)$的上界，不确定性 $\Delta f(\boldsymbol{x},t)$由下式约束：

$$|\Delta f(\boldsymbol{x},t)|\leqslant F(\boldsymbol{x},t) \tag{2.58}$$

2.10.2　滑模控制器设计

系统的滑模平面为

$$s(t)=cx_1(t)+x_2(t) \tag{2.59}$$

其中 c 为正常数。

设计控制律为

$$u(t)=\frac{1}{a}[-f(\boldsymbol{x},t)-cx_2(t)-k(\boldsymbol{x},t)\operatorname{sgn}(s(t))] \tag{2.60}$$

式中 $k(\boldsymbol{x},t)$项为控制增益：

$$k(\boldsymbol{x},t)=F(\boldsymbol{x},t)+D+\eta \tag{2.61}$$

其中 $\eta>0$。

稳定性分析：

对滑模面式(2.59)求导得

$$\dot{s}(t)=cx_2(t)+\dot{x}_2(t)=cx_2(t)+f(\boldsymbol{x},t)+\Delta f(\boldsymbol{x},t)+au(t)+d(t) \tag{2.62}$$

将 $u(t)$ 代入式(2.62)得

$$\dot{s}(t)=\Delta f(\boldsymbol{x},t)+d(t)-k(\boldsymbol{x},t)\operatorname{sgn}(s(t)) \tag{2.63}$$

则

$$s(t)\dot{s}(t)=(d(t)+\Delta f(\boldsymbol{x},t))s(t)-k(\boldsymbol{x},t)\mid s(t)\mid \tag{2.64}$$

将式(2.61)代入式(2.64)得

$$s(t)\dot{s}(t)\leqslant-\eta\mid s(t)\mid$$

即

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}(s^2(t))\leqslant-\eta\mid s(t)\mid$$

为减少抖振，可采用饱和函数或继电特性来连续化控制量，即用饱和函数或继电特性函数代替式(2.60)中的开关特性函数。

2.10.3 仿真实例

考虑二阶系统，误差状态方程为

$$\begin{cases}\dot{x}_1(t)=x_2(t)\\ \dot{x}_2(t)=f(\boldsymbol{x},t)+\Delta f(\boldsymbol{x},t)+au(t)+d(t)\end{cases}$$

其中 $a=5.0$，$f(\boldsymbol{x},t)=3x_2(t)$，$\Delta f(\boldsymbol{x},t)=3\sin(t)$，$d(t)=10\sin(t)$。

取 $F(\boldsymbol{x},t)=3.0$，$D=10$，$\eta=10$。则 $k(\boldsymbol{x},t)=F(\boldsymbol{x},t)+D+\eta=23$。

将误差状态方程写为位置状态方程的形式：

$$\begin{cases}\dot{y}_1(t)=y_2(t)\\ \dot{y}_2(t)=\ddot{r}(t)-3(\dot{r}(t)-y_2(t))-\Delta f(x,t)-au(t)-d(t)\end{cases}$$

取位置跟踪信号为 $r(t)=\sin(2\pi t)$，则 $x_1(t)=r(t)-y_1(t)$，$x_2(t)=\dot{r}(t)-y_2(t)$，控制器参数取 $c=5$。对象初始状态取[-0.15，-0.15]，采用控制律式(2.60)，$M=1,2,3$ 分别表示切换函数、饱和函数和继电特性函数方法。取 $M=1$，仿真结果如图 2-44～图 2-46 所示。

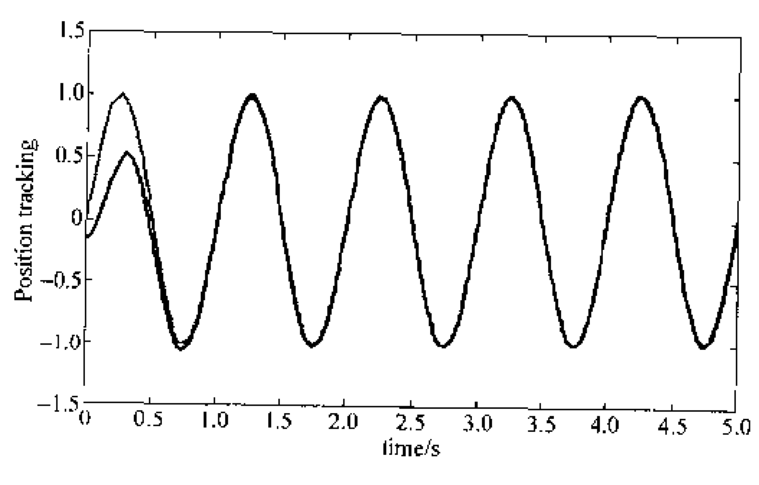


图 2-44 位置跟踪

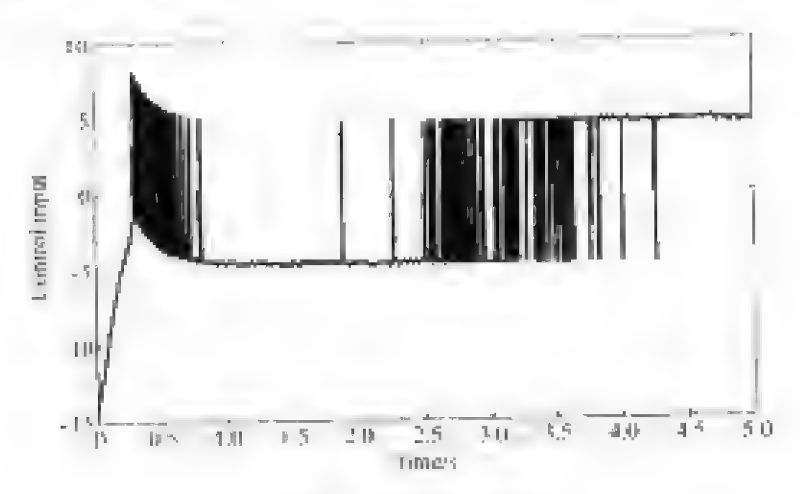


图2-45 控制输入信号

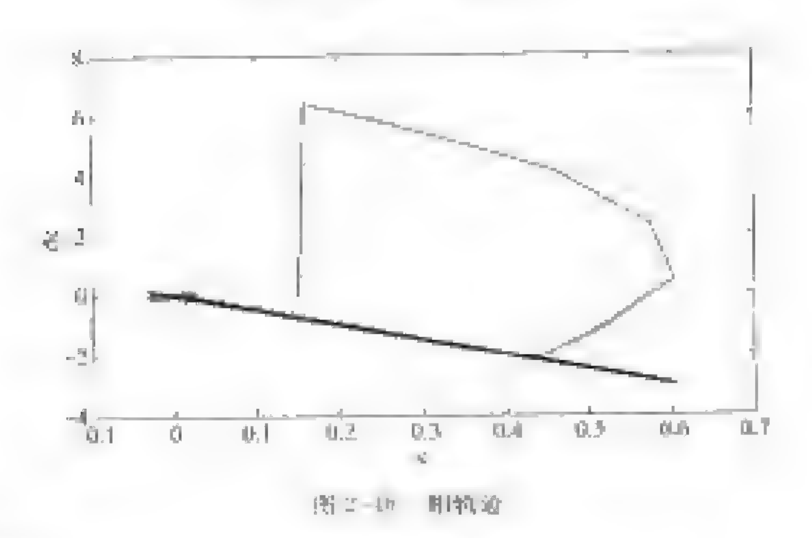


图2-46 相轨迹

仿真程序如下。

(1) Simulink 主程序(如图2-47所示)：chap2_8sim.mdl

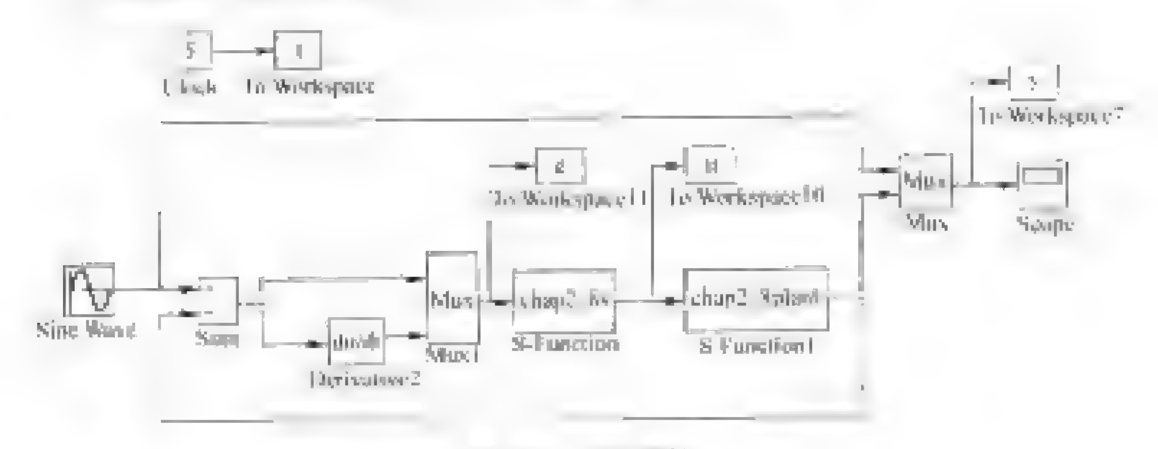


图2-47 主程序图

(2) 控制器S函数：chap2_8s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
```

```
switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [];

function sys = mdlOutputs(t,x,u)
r = sin(2 * pi * t);
dr = 2 * pi * cos(2 * pi * t);
ddr = - (2 * pi)^2 * sin(2 * pi * t);

x1 = u(1);
x2 = u(2);

a = 5.0;
fx = 3 * x2;

c = 5;
s = c * x1 + x2;

F = 3.0;
D = 10;
xite = 10;
kx = F + D + xite;

M = 1;
if M == 1              % Switch function
   ut = 1/a * ( - fx - c * x2 - kx * sign(s));
elseif M == 2          % Saturated function
   fai = 0.20;
   if abs(s) <= fai
      sat = s/fai;
   else
```

```
        sat = sign(s);
    end
    ut = 1/a * ( - fx - c * x2 - kx * sat);
elseif M == 3          % Relay function
    delta = 0.001;
    rs = s/(abs(s) + delta);
    ut = 1/a * ( - fx - c * x2 - kx * rs);
end

sys(1) = ut;
```

(3) 被控对象 S 函数：chap2_8plant.m

```
% S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [ - 0.15   - 0.15];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
a = 5.0;
dt = 10 * sin(t);
```

```
r = sin(2 * pi * t);
dr = 2 * pi * cos(2 * pi * t);
ddr = - (2 * pi)^2 * sin(2 * pi * t);

x1 = r - x(1);
x2 = dr - x(2);

fx = 3 * x2;
df = 3 * sin(t);
sys(1) = x(2);
sys(2) = ddr - fx - df - a * u - dt;

function sys = mdlOutputs(t,x,u)

sys(1) = x(1);
```

(4) 作图程序：chap2_8plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

c = 5;
figure(3);
plot(e(:,1),e(:,2),'r',e(:,1),-c'.*e(:,1),'b');
xlabel('e');ylabel('de');
```

参考文献

1. Utkin V I, Guldner J, Shi J. Sliding mode control in electromechanical systems. Taylor & Francis, 1999
2. Liu Jinkun, Er Lianjie. Sliding Mode Controller Design for Position and Speed Control of Flight Simulator Servo System with Large Friction. Systems Engineering and Electronics(China), 2003, 14(3): 59～62
3. 刘强，尔联洁，刘金琨. 摩擦非线性环节的特性、建模与控制补偿综述. 系统工程与电子技术，2002, 24(11): 45～52
4. 刘强，尔联洁，刘金琨. 参数不确定机械伺服系统的鲁棒非线性摩擦补偿控制. 自动化学报，2003, 29(4): 628～632
5. 刘金琨. 先进 PID 控制及其 MATLAB 仿真(第 2 版). 北京：电子工业出版社，2004

第 3 章　离散时间系统滑模控制

从理论上说，滑模变结构控制主要是针对连续系统模型。因为只有理想的连续滑模变结构控制，才具有切换逻辑变结构控制产生的等效控制 u_{eq}。对于离散系统，滑模变结构控制不能产生理想的滑动模态，只能产生准滑模控制[1]。在实际工程中，计算机实时控制均为离散系统，离散系统滑模变结构控制的研究与设计成为滑模变结构控制理论与应用的一个重要组成部分。在 20 世纪 80 年代后期，离散滑模变结构控制迅速发展起来，并在工程领域得到了一系列的应用。

3.1　离散滑模控制描述

离散系统状态方程为

$$x(k+1) = Ax(k) + Bu(k) \tag{3.1}$$

切换函数设计为

$$s(k) = Cx(k) \tag{3.2}$$

其中 $C=[c_1, c_2, \cdots, c_n]$，$c_n=1.0$。

连续滑模变结构控制中讨论的三个基本问题（滑动模态的存在性、可达性及稳定性）也是离散时间系统的基本问题。但是由于离散控制的固有特点，这些问题以及滑模变结构控制策略的表达形式与连续系统不同。

3.2　离散时间滑模控制的特性

3.2.1　准滑动模态

定义一个包围切换面的切换带

$$S^{\Delta} = \{x \in \mathbf{R}^n \mid -\Delta < s(x) = cx < +\Delta\} \tag{3.3}$$

从任意初始状态出发的离散系统的运动，或者于有限步到达切换面 s，然后在其上运动，称之为理想准滑动模态；或者在带内运动，步步穿越切换面，称之为非理想准滑动模态。系统发生在切换带内的两种准滑动模态，称为离散变结构控制的准滑动模态。2Δ 称为切换带的宽度，如图 3-1 所示。

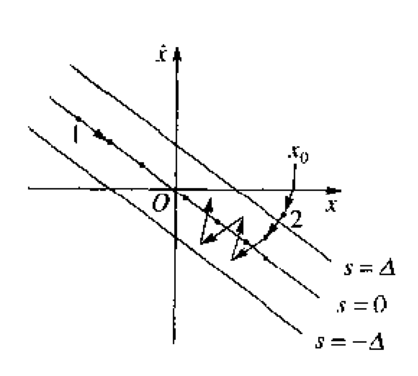


图 3-1　准滑动模态

离散滑模变结构控制中，从任意状态出发的运动可分为以下三个阶段：

（1）趋近模态：从初始状态趋向切换带。

（2）准滑动模态：或为理想的，或为非理想的准滑动模态。

（3）平稳状态：或为原点 $x=0$，属于理想准滑动模态情况；或为围绕原点的抖振，属于非理想准滑动模态。

离散滑模变结构控制中，从任意状态出发的运动应满足如下特性：

（1）运动从任意初始条件出发，单调地向切换面趋近，并在有限步骤内到达或穿越切换面。

（2）运动一旦穿越切换面，它的每一个后续步骤均从另一面穿越切换面，并一直进行下去。

（3）穿越开始后，每一步的长度是非递增的，运动轨迹限于一特定带内。

（4）平稳状态或为原点 $x=0$，即属于理想准滑动模态情况；或为围绕原点的抖振，即属于非理想准滑动模态。

3.2.2　离散滑模的存在性和可达性

把连续系统的到达条件推广到离散系统，到达条件为

$$[s(k+1)-s(k)]s(k)<0 \tag{3.4}$$

到达条件式(3.4)对于准滑模运动的存在是必要条件，而不是充分条件，并不能保证系统的稳定性。例如，当离散系统运动轨迹围绕切换面 $s=0$ 作幅值发散的振荡时，式(3.4)也能满足。

选取李雅普诺夫函数

$$V(k)=\frac{1}{2}s^2(k) \tag{3.5}$$

只要满足条件

$$\Delta V(k)=s^2(k+1)-s^2(k)<0,\quad s(k)\neq 0 \tag{3.6}$$

则根据李雅普诺夫稳定性定理，$s(k)=0$ 是全局渐近稳定的平衡面，即任意初始位置的状态都会趋向于切换面 $s(k)$。所以取到达条件为

$$s^2(k+1)<s^2(k) \tag{3.7}$$

当采样时间 T 很小时，离散滑模的存在和到达性条件为[4]

$$[s(k+1)-s(k)]\mathrm{sgn}(s(k))<0,\quad [s(k+1)+s(k)]\mathrm{sgn}(s(k))>0 \tag{3.8}$$

3.2.3　离散滑模控制的不变性

考虑如下受干扰及参数摄动的离散系统：

$$\left.\begin{aligned}&\boldsymbol{x}(k+1)=\boldsymbol{G}\boldsymbol{x}(k)+\Delta\boldsymbol{G}\boldsymbol{x}(k)+\boldsymbol{B}\boldsymbol{u}(k)+\boldsymbol{D}\boldsymbol{f}(k)\\&\boldsymbol{x}(k)\in\mathbf{R}^n,\ \boldsymbol{u}(k)\in\mathbf{R}^m,\ \boldsymbol{f}(k)\in\mathbf{R}^l\end{aligned}\right\} \tag{3.9}$$

其中 $\Delta\boldsymbol{G}\boldsymbol{x}(k)$ 表示系统参数的摄动，$\boldsymbol{D}\boldsymbol{f}(k)$ 表示系统所受外干扰的影响。

假设摄动与干扰满足匹配条件：

$$\Delta\boldsymbol{G}=\boldsymbol{B}\widetilde{\boldsymbol{G}},\qquad \boldsymbol{D}=\boldsymbol{B}\widetilde{\boldsymbol{D}} \tag{3.10}$$

其中 $\tilde{G}$,$\tilde{D}$ 是不确定的。系统式(3.9)可写成

$$\left.\begin{array}{l} \boldsymbol{x}(k+1)=\boldsymbol{G}\boldsymbol{x}(k)+\boldsymbol{B}[\tilde{\boldsymbol{G}}\boldsymbol{x}(k)+u(k)+\tilde{\boldsymbol{D}}f(k)] \\ \boldsymbol{x}(k)\in\mathbf{R}^{n},\ u(k)\in\mathbf{R}^{m},\ f(k)\in\mathbf{R}^{l} \end{array}\right\} \tag{3.11}$$

由式(3.11)可以看出,离散系统和连续系统一样,对系统的参数摄动及外干扰是不变的,其充分必要条件为式(3.10)。

3.3 基于趋近律的离散滑模控制

3.3.1 离散趋近律的设计

趋近律方法是滑模变结构控制的一种典型控制策略,这种控制方法不仅可以对系统在切换面附近或沿切换面的滑模运动段进行分析,而且可以有效地对系统趋近段的动态过程进行分析和设计,从而保证系统在整个状态空间内具有良好的运动品质。

对于连续滑模变结构控制,常用的趋近律为指数趋近律

$$\dot{s}(t)=-\varepsilon\operatorname{sgn}(s(t))-qs(t) \qquad \varepsilon>0,q>0 \tag{3.12}$$

相应地可以设计离散指数趋近律。针对离散系统

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}u(k)$$

将式(3.12)离散化,得到指数趋近律为

$$\frac{s(k+1)-s(k)}{T}=-\varepsilon\operatorname{sgn}(s(k))-qs(k)$$

$$s(k+1)-s(k)=-qTs(k)-\varepsilon T\operatorname{sgn}(s(k)) \tag{3.13}$$

其中 $\varepsilon>0,q>0,1-qT>0$,T 为采样周期。

高为炳[1]对离散趋近律式(3.13)进行了分析,得到以下六点结论:

(1) 运动单调地向切换面趋近;

(2) 运动从任意初始状态在有限步到达或穿越切换面;

(3) 运动一旦穿越过切换面,它的每一个后继步均从另一面穿越切换面,并一直进行下去;

(4) 不应发生运动渐近地趋向切换面而不穿越的情况;

(5) 不应发生运动从初始状态开始每一步均来回地穿越切换面的情况;

(6) 不应发生一旦运动开始穿越切换面,每一步的长度不断增长的情况。

由于基于指数的离散趋近律式(3.13)满足,则

$$\begin{aligned}[s(k+1)-s(k)]\operatorname{sgn}(s(k))&=[-qTs(k)-\varepsilon T\operatorname{sgn}(s(k))]\operatorname{sgn}(s(k))\\&=-qT\mid s(k)\mid-\varepsilon T\mid s(k)\mid<0\end{aligned}$$

同时,当采样时间 T 很小时,$2-qT\gg0$,有

$$\begin{aligned}[s(k+1)+s(k)]\operatorname{sgn}(s(k))&=[(2-qT)s(k)-\varepsilon T\operatorname{sgn}(s(k))]\operatorname{sgn}(s(k))\\&=(2-qT)\mid s(k)\mid-\varepsilon T\mid s(k)\mid>0\end{aligned}$$

所以,离散趋近律式(3.13)满足到达条件,可保证趋近律模态具有良好的品质,并且切换带的大小可以计算,求解滑动模态控制直接而简单。

3.3.2 离散控制律的设计

离散滑模面为

$$s(k) = \boldsymbol{C}\boldsymbol{x}(k)$$

将 $s(k+1)=\boldsymbol{C}\boldsymbol{x}(k+1)=\boldsymbol{CA}\boldsymbol{x}(k)+\boldsymbol{CB}u(k)$代入趋近律得

$$-(Tq-1)s(k)-\varepsilon T\operatorname{sgn}(s(k)) = \boldsymbol{CA}\boldsymbol{x}(k)+\boldsymbol{GB}u(k)$$

假设滑模变结构可控条件 $\boldsymbol{CB}\neq 0$ 成立，离散滑模控制律为

$$u(k) = -(\boldsymbol{CB})^{-1}[\boldsymbol{CA}\boldsymbol{x}(k)-(1-qT)s(k)+\varepsilon T\operatorname{sgn}(s(k))] \tag{3.14}$$

为了防止控制器发生抖振，可采用饱和函数 sat(s)代替理想滑动模态中的符号函数 sgn(s)：

$$\operatorname{sat}(s) = \begin{cases} 1 & s > \Delta \\ ks & |s| \leqslant \Delta \quad k=\dfrac{1}{\Delta} \\ -1 & s < -\Delta \end{cases} \tag{3.15}$$

则控制律式(3.14)变为

$$u(k) = -(\boldsymbol{CB})^{-1}[\boldsymbol{CA}\boldsymbol{x}(k)-(1-qT)s(k)+\varepsilon T\operatorname{sat}(s)] \tag{3.16}$$

3.3.3 仿真实例

针对二阶离散系统

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}u(k)$$

其中 $\boldsymbol{A}=\begin{bmatrix} 1 & 0.001 \\ 0 & 0.9753 \end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix} -0.0001 \\ -0.1314 \end{bmatrix}$。

采样时间为 1ms，初始状态为 $\boldsymbol{x}(0)=[0.5,\ 0.5]^{\mathrm{T}}$。取 $C=5$，$q=10$。$M=1$ 时，采用控制律式(3.14)，取 $\varepsilon=0.50$，仿真结果如图 3-2～图 3-5 所示；$M=2$ 时，采用带有饱和函数的式(3.16)，取 $\Delta=0.005$，仿真结果如图 3-6 和图 3-7 所示。可见，采用饱和函数后，控制器输出的抖振大大下降。

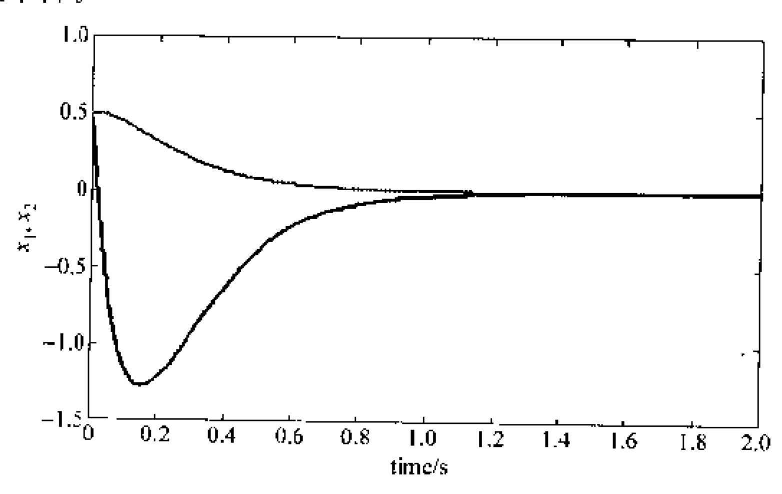


图 3-2 状态变量的变化曲线（$M=1$）

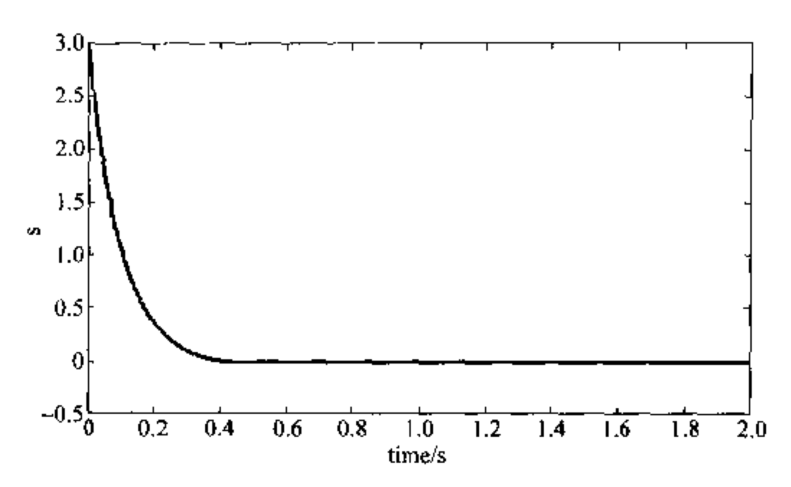


图 3-3　切换函数的变化曲线（$M=1$）

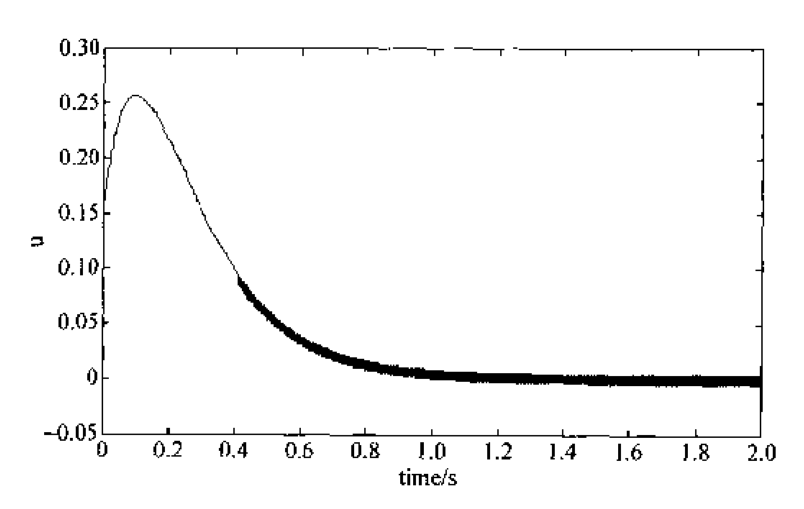


图 3-4　控制器输出变化曲线（$M=1$）

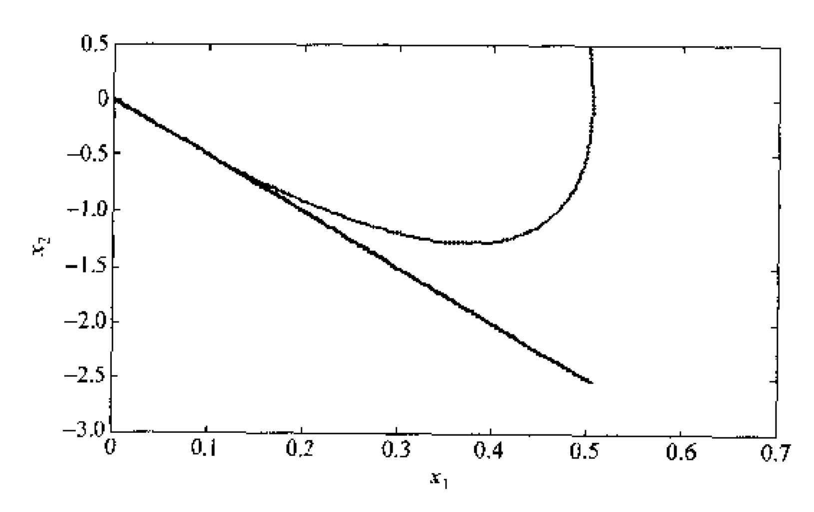


图 3-5　滑模运动相轨迹（$M=1$）

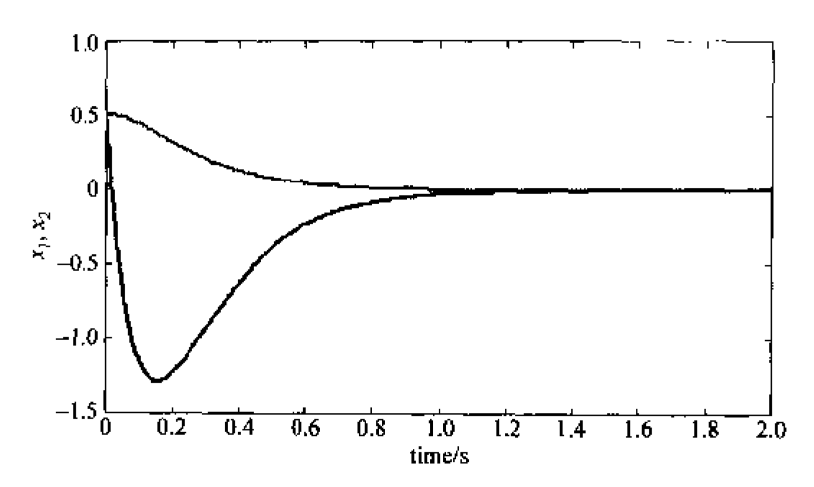


图 3-6　状态变量的变化曲线（$M=2$）

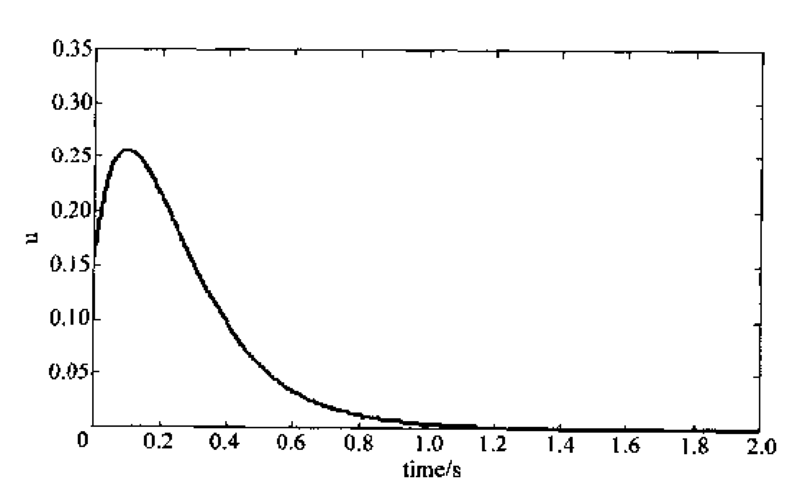


图 3-7　控制器输出变化曲线（$M=2$）

仿真程序：chap3_1.m

```
clear all;
close all;

A=[1 0.0010;
   0 0.9753];
B=[-0.0001;
   -0.1314];
x=[0.5;0.5];

ts=0.001;
for k=1:1:2000
time(k)=k*ts;

c=5;q=10;ep=0.5;
C=[c 1];
s(k)=C*x;
```

```
M = 2;
if M == 1
    u(k) = - inv(C * B) * (C * A * x - (1 - q * ts) * s(k) + ep * ts * sign(s(k)));
elseif M == 2            % Saturated function
    delta = 0.005;
    kk = 1/delta;
    if s(k)>delta
        sats = 1;
    elseif abs(s(k))< = delta
        sats = kk * s(k);
    elseif s(k)< - delta
        sats = - 1;
    end
    u(k) = - inv(C * B) * (C * A * x - (1 - q * ts) * s(k) + ep * ts * sats);
end

x = A * x + B * u(k);

x1(k) = x(1);
x2(k) = x(2);
end
figure(1);
plot(time,x1,'r',time,x2,'b');
xlabel('time(s)');ylabel('x1,x2');
figure(2);
plot(time,s,'r');
xlabel('time(s)');ylabel('s');
figure(3);
plot(time,u,'r');
xlabel('time(s)');ylabel('u');
figure(4);
plot(x1,x2,'r',x1, - c * x1,'b');
xlabel('x1');ylabel('x2');
```

3.4　基于等效控制的离散滑模控制

3.4.1　控制器设计

对离散系统

$$x(k+1) = \boldsymbol{A}x(k) + \boldsymbol{B}u(k) \qquad \boldsymbol{x} \in \mathbf{R}^n,\ u \in \mathbf{R} \tag{3.17}$$

取切换函数

$$s(k) = \boldsymbol{C}_e\boldsymbol{x}(k) \tag{3.18}$$

离散系统进入理想滑动模态时，$s(k)$满足

$$s(k+1)=s(k) \tag{3.19}$$

即

$$s(k+1)=C_e x(k+1)=C_e Ax(k)+C_e Bu(k) \tag{3.20}$$

可得等效控制为

$$u_{eq}=-(C_e B)^{-1}C_e^{T}(A-I)x(k) \tag{3.21}$$

K. Furuta 提出了一种离散滑模变结构控制律[2]，采用以等效控制为基础的形式：

$$u(k)=u_{eq}+F_D x(k) \tag{3.22}$$

式中 $F_D=[f_1,f_2,\cdots,f_n]$，F_D 中的各个元素表示系统各状态变量的增益。

稳定性证明：

定义 Lyapunov 函数

$$V(k)=\frac{1}{2}s(k)^2 \tag{3.23}$$

由式(3.18)得

$$s(k)=C_e x(k)$$
$$s(k+1)=C_e x(k+1)=C_e Ax(k)+C_e Bu(k) \tag{3.24}$$

将控制律式(3.21)、式(3.22)代入式(3.24)，得

$$s(k+1)=C_e Ax(k)-C_e(A-I)x(k)+C_e BF_D x(k)=C_e x(k)+C_e BF_D x(k)$$

则

$$s(k+1)^2-s(k)^2=2s(k)C_e BF_D x(k)+(C_e BF_D x(k))^2$$

要使 $s(k+1)^2-s(k)^2<0$，只要

$$(C_e BF_D x(k))^2<-2s(k)C_e BF_D x(k)$$

即对于每一个 i，有

$$\frac{1}{2}(C_e B)^2\left|f_i\right|\left|x_i(k)\right|\sum_{j=1}^{n}\left|f_j\right|\left|x_j(k)\right|<-s(k)C_e Bf_i x_i(k) \tag{3.25}$$

定义

$$\delta_i=\frac{1}{2}f_0(C_e B)^2\left|x_i(k)\right|\sum_{j=1}^{n}\left|x_j(k)\right| \tag{3.26}$$

其中，$f_0>0$，$|f_i|=f_0$，$|f_j|=f_0$。

将式(3.26)代入式(3.25)，得

$$\delta_i f_0<-C_e Bs(k)f_i x_i(k) \tag{3.27}$$

因此

$$f_i=\begin{cases}f_0 & C_e Bs(k)x_i(k)<-\delta_i\\-f_0 & C_e Bs(k)x_i(k)>\delta_i\end{cases} \tag{3.28}$$

由式(3.27)可知

$$|s(k)|>|C_e Bx_i(k)|^{-1}\delta_i \tag{3.29}$$

可见，离散到达条件式(3.7)只能保证在式(3.29)条件下成立。

另外，由式(3.26)和式(3.29)得

$$f_0 < \frac{2\,|\,s(k)\,|}{|\,\boldsymbol{C}_e\boldsymbol{B}\,|\,\sum\limits_{i=1}^{n}|\,x_i(k)\,|} \tag{3.30}$$

综上所述，针对离散系统式(3.17)，基于等效控制的离散滑模控制律为

$$u(k) = u_{eq} + \boldsymbol{F}_D\boldsymbol{x}(k)$$

其中 $\boldsymbol{F}_D=[f_1,f_2,\cdots,f_n]$。

$$u_{eq} = -(\boldsymbol{C}_e\boldsymbol{B})^{-1}\boldsymbol{C}_e^T(\boldsymbol{A}-\boldsymbol{I})\boldsymbol{x}(k)$$

$$f_i = \begin{cases} f_0 & \boldsymbol{C}_e\boldsymbol{B}s(k)x_i(k) < -\delta_i \\ 0 & -\delta_i \leqslant \boldsymbol{C}_e\boldsymbol{B}s(k)x_i(k) \leqslant \delta_i \\ -f_0 & \boldsymbol{C}_e\boldsymbol{B}s(k)x_i(k) > \delta_i \end{cases}$$

$$\delta_i = \frac{1}{2}f_0(\boldsymbol{C}_e\boldsymbol{B})^2\,|\,x_i(k)\,|\,\sum_{i=1}^{n}|\,x_i(k)\,|$$

$$0 < f_0 < \frac{2\,|\,s(k)\,|}{|\,\boldsymbol{C}_e\boldsymbol{B}\,|\,\sum\limits_{i=1}^{n}|\,x_i(k)\,|}$$

3.4.2 仿真实例

对象传递函数为

$$G(s) = \frac{133}{s(s+25)}$$

采样时间为1ms，采用Z变换进行离散化，离散状态方程为

$$\begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}u(k) \\ y(k+1) = \boldsymbol{C}x(k) + D \end{cases}$$

其中 $\boldsymbol{A}=\begin{bmatrix}1 & 0.0010\\0 & 0.9753\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0.0001\\0.1314\end{bmatrix}$，$\boldsymbol{C}=[1,0]$，$D=0$。

设被控对象的初值为 $\boldsymbol{x}=[0.5,0.5]^T$。采用基于等效控制的离散滑模控制律式(3.22)，取 $\boldsymbol{C}_e=[20,1]$，仿真结果如图3-8～图3-10所示。

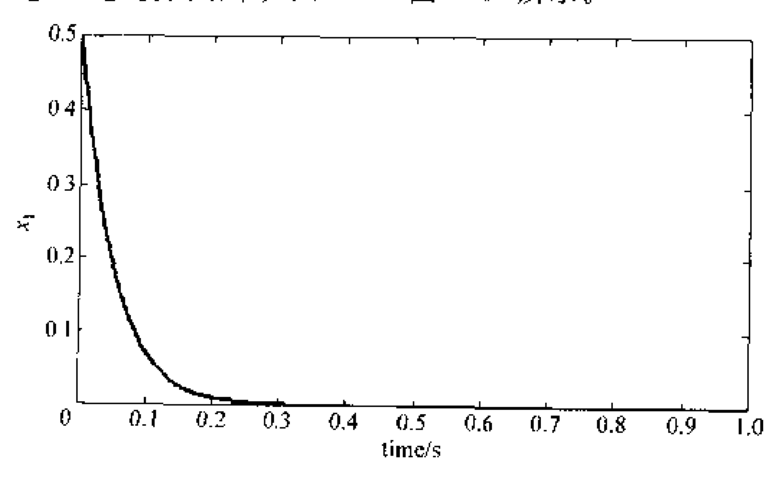


图3-8 状态变量的变化曲线

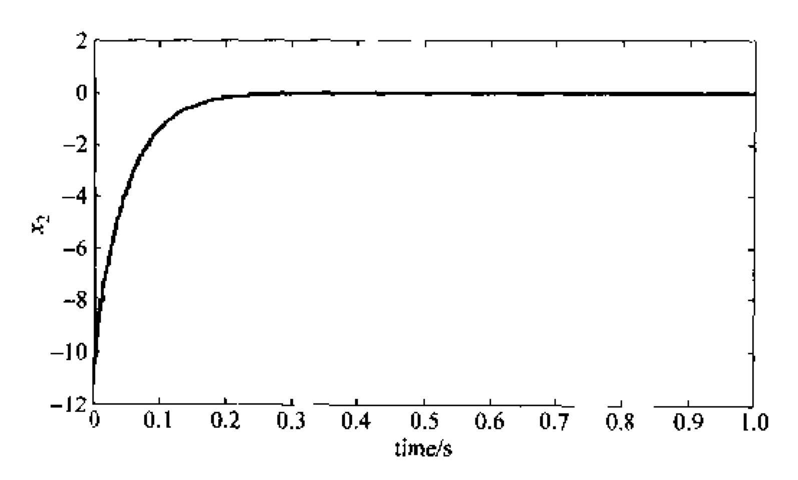


图 3-9 状态变量的变化曲线

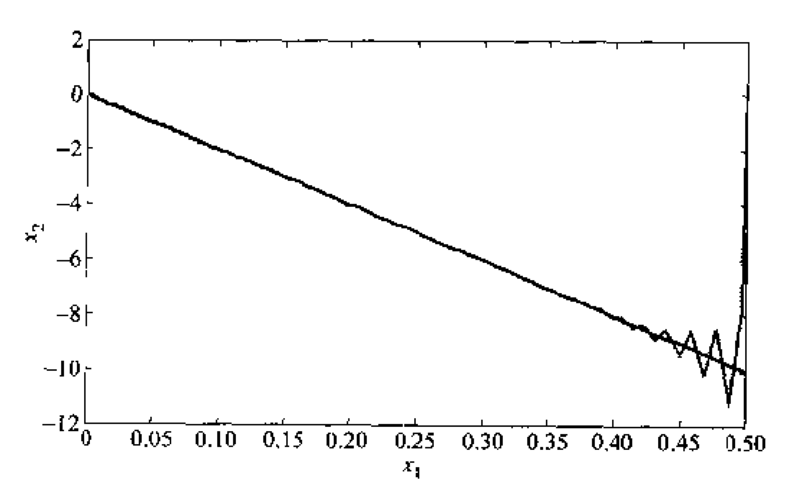


图 3-10 滑模运动的相轨迹

仿真程序：chap3_2.m

```
clear all;
close all;

a=25;b=133;
A1=[0,1;0,-a];
B1=[0;b];
C1=[1,0];
D1=0;

ts=0.001;
[A,B,C,D]=c2dm(A1,B1,C1,D1,ts,'z');

c=20;
Ce=[c,1];
x=[0.5;0.5];

ts=0.001;
for k=1:1:1000
time(k)=k*ts;
```

```
    s(k) = Ce * x;
    x1(k) = x(1);
    x2(k) = x(2);

    f0 = 2 * abs(s(k))/(abs(Ce * B) * (abs(x1(k)) + abs(x2(k))) + 0.2);
    deta1 = 0.5 * f0 * (Ce * B) * (Ce * B) * abs(x1(k)) * (abs(x1(k)) + abs(x2(k)));
    deta2 = 0.5 * f0 * (Ce * B) * (Ce * B) * abs(x2(k)) * (abs(x1(k)) + abs(x2(k)));

    cond1 = Ce * B * s(k) * x1(k);
if cond1 < - deta1
    f1 = f0;
elseif abs(cond1) <= deta1
    f1 = 0;
elseif cond1 > deta1
    f1 = - f0;
end

cond2 = Ce * B * s(k) * x2(k);
if cond2 < - deta2
    f2 = f0;
elseif abs(cond2) <= deta2
    f2 = 0;
elseif cond2 > deta2
    f2 = - f0;
end

Fd = [f1,f2];
ueq(k) = - 1/(Ce * B) * Ce * (A - eye(2)) * x;
u(k) = ueq(k) + Fd * x;

x = A * x + B * u(k);

end
figure(1);
plot(time,x1,'r');
xlabel('time(s)');ylabel('x1');
figure(2);
plot(time,x2,'r');
xlabel('time(s)');ylabel('x2');
figure(3);
plot(time,u,'r');
xlabel('time(s)');ylabel('u');
figure(4);
plot(x1,x2,'r',x1, - c * x1,'b');
xlabel('x1');ylabel('x2');
```

3.4.3 位置跟踪控制器的设计

对离散系统

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}u(k) \qquad x \in \mathbf{R}^n,\ u \in \mathbf{R} \tag{3.31}$$

分析离散系统的跟踪问题，首先要将它转化为误差状态方程。假设 $\boldsymbol{R}(k)$ 为希望的指令信号，设 $\boldsymbol{x}_e(k)=e(k)=\boldsymbol{R}(k)-\boldsymbol{x}(k)$，则可得系统式(3.31)的离散误差状态方程为

$$\boldsymbol{x}_e(k+1)=\boldsymbol{A}_e\boldsymbol{x}_e(k)+\boldsymbol{B}_e u(k)+f(k) \tag{3.32}$$

其中

$$\boldsymbol{A}_e=\boldsymbol{A},\boldsymbol{B}_e=-\boldsymbol{B},\quad f(k)=-\boldsymbol{AR}(k)+\boldsymbol{R}(k+1) \tag{3.33}$$

切换函数取为

$$s_e(k)=\boldsymbol{C}_e\boldsymbol{x}_e(k) \tag{3.34}$$

由 $s_e(k)=s_e(k+1)$ 得等效控制律为

$$u_{eq}(k)=-(\boldsymbol{C}_e\boldsymbol{B}_e)^{-1}[\boldsymbol{C}_e(\boldsymbol{A}_e-\boldsymbol{I})\boldsymbol{x}_e(k)+\boldsymbol{C}_e f(k)] \tag{3.35}$$

总控制律设计为

$$u(k)=u_{eq}+\boldsymbol{F}_D\boldsymbol{x}_e(k) \tag{3.36}$$

式中 $\boldsymbol{F}_D=[f_1,f_2,\cdots,f_n]$，$\boldsymbol{F}_D$ 中的各个元素表示系统各状态变量的增益。

稳定性证明：

定义 Lyapunov 函数

$$V(k)=\frac{1}{2}s_e^2(k) \tag{3.37}$$

由已知得

$$s_e(k)=\boldsymbol{C}_e\boldsymbol{x}_e(k)$$

$$s_e(k+1)=\boldsymbol{C}_e\boldsymbol{x}_e(k+1)=\boldsymbol{C}_e\boldsymbol{A}_e\boldsymbol{x}_e(k)+\boldsymbol{C}_e\boldsymbol{B}_e u(k)+\boldsymbol{C}_e f(k)$$

将控制律式(3.22)代入上式，得

$$\begin{aligned}s_e(k+1)&=\boldsymbol{C}_e\boldsymbol{A}_e\boldsymbol{x}_e(k)+\boldsymbol{C}_e\boldsymbol{B}_e u(k)+\boldsymbol{C}_e f(k)\\&=\boldsymbol{C}_e\boldsymbol{A}_e\boldsymbol{x}_e(k)+\boldsymbol{C}_e\boldsymbol{B}_e\{-(\boldsymbol{C}_e\boldsymbol{B}_e)^{-1}[\boldsymbol{C}_e(\boldsymbol{A}_e-\boldsymbol{I})\boldsymbol{x}_e(k)+\boldsymbol{C}_e f(k)]+\boldsymbol{F}_D\boldsymbol{x}_e(k)\}+\boldsymbol{C}_e f(k)\\&=(\boldsymbol{C}_e+\boldsymbol{C}_e\boldsymbol{B}_e\boldsymbol{F}_D)\boldsymbol{x}_e(k)\end{aligned}$$

所以

$$\Delta s_e(k+1)=s_e(k+1)-s_e(k)=\boldsymbol{C}_e\boldsymbol{B}_e\boldsymbol{F}_D\boldsymbol{x}_e(k)$$

$$s_e(k)\Delta s_e(k+1)=s_e(k)\boldsymbol{C}_e\boldsymbol{B}_e\boldsymbol{F}_D\boldsymbol{x}_e(k)$$

由于

$$\boldsymbol{F}_D\boldsymbol{x}_e(k)=\sum_{i=1}^{n}f_i x_{ei}(k)$$

定义

$$\delta_i=\frac{1}{2}f_0(\boldsymbol{C}_e\boldsymbol{B}_e)^2\mid x_{ei}(k)\mid\sum_{i=1}^{n}\mid x_{ei}(k)\mid \tag{3.38}$$

及

$$f_i=\begin{cases}f_0 & \boldsymbol{C}_e\boldsymbol{B}_e s_e(k)x_{ei}(k)<-\delta_i\\0 & -\delta_i\leqslant\boldsymbol{C}_e\boldsymbol{B}_e s_e(k)x_{ei}(k)\leqslant\delta_i\\-f_0 & \boldsymbol{C}_e\boldsymbol{B}_e s_e(k)x_{ei}(k)>\delta_i\end{cases} \tag{3.39}$$

则

$$\begin{aligned}s_e(k)\Delta s_e(k+1)&<-\sum_{i=1}^{n}\delta_i\mid f_i\mid\leqslant-\frac{1}{2}(\boldsymbol{C}_e\boldsymbol{B}_e)^2\Big(\sum_{i=1}^{n}\mid f_i\mid\mid x_{ei}(k)\mid\Big)^2\\&\leqslant-\frac{1}{2}(\Delta s_e(k+1))^2\ \frac{1}{2}(\Delta s_e(k+1))^2+s_e(k)\Delta s_e(k+1)<0\end{aligned} \tag{3.40}$$

由于

$$
\begin{aligned}
V(k+1) &= \frac{1}{2}s_e^2(k+1) = \frac{1}{2}(s_e(k) + \Delta s_e(k+1))^2 \\
&= \frac{1}{2}s_e^2(k) + \frac{1}{2}\Delta s_e^2(k+1) + s_e(k)\Delta s_e(k+1) \\
&= V(k) + \frac{1}{2}\Delta s_e^2(k+1) + s_e(k)\Delta s_e(k+1)
\end{aligned}
$$

由式(3.40)可得

$$V(k+1) < V(k)$$

可见，控制器式(3.36)满足到达条件。

综上所述，针对离散系统式(3.31)，基于等效控制的离散滑动模态控制律为

$$
\begin{aligned}
& u(k) = u_{eq} + \boldsymbol{F}_D \boldsymbol{x}_e(k) \\
& u_{eq}(k) = -(\boldsymbol{C}_e\boldsymbol{B}_e)^{-1}[\boldsymbol{C}_e(\boldsymbol{A}_e - \boldsymbol{I})\boldsymbol{x}_e(k) + \boldsymbol{C}_e f(k)] \\
& \boldsymbol{F}_D = [f_1, f_2, \cdots, f_n] \\
& f_i = \begin{cases} f_0 & \boldsymbol{C}_e\boldsymbol{B}_e s_e(k) x_{ei}(k) < -\delta_i \\ 0 & -\delta_i \leqslant \boldsymbol{C}_e\boldsymbol{B}_e s_e(k) x_{ei}(k) \leqslant \delta_i \\ -f_0 & \boldsymbol{C}_e\boldsymbol{B}_e s_e(k) x_{ei}(k) > \delta_i \end{cases} \\
& \delta_i = \frac{1}{2} f_0 (\boldsymbol{C}_e\boldsymbol{B}_e)^2 \mid x_{ei}(k) \mid \sum_{i=1}^{n} \quad x_{ei}(k) \mid \\
& 0 < f_0 < \frac{2 \mid s_e(k) \mid}{\mid \boldsymbol{C}_e\boldsymbol{B}_e \mid \sum_{i=1}^{n} \mid x_{ei}(k) \mid}
\end{aligned}
$$

3.4.4　仿真实例

对象传递函数为

$$G(s) = \frac{133}{s(s+25)}$$

采样时间为 1ms，采用 Z 变换进行离散化，离散状态方程为

$$
\begin{cases}
\boldsymbol{x}(k+1) = \boldsymbol{A}_2 \boldsymbol{x}(k) + \boldsymbol{B}_2 u(k) \\
y(k+1) = \boldsymbol{C}_2 \boldsymbol{x}(k) + D_2
\end{cases}
$$

其中 $\boldsymbol{A}_2 = \begin{bmatrix} 1 & 0.0010 \\ 0 & 0.9753 \end{bmatrix}$，$\boldsymbol{B}_2 = \begin{bmatrix} 0.0001 \\ 0.1314 \end{bmatrix}$，$\boldsymbol{C}_2 = [1, 0]$，$D_2 = 0$。

令指令为 $r(k)$，其变化率为 $\mathrm{d}r(k)$。取 $\boldsymbol{R} = [r(k), \mathrm{d}r(k)]^T$，$\boldsymbol{R}_1 = [r(k+1), \mathrm{d}r(k+1)]^T$。将离散状态方程转化为离散误差状态方程为

$$
\begin{aligned}
& \boldsymbol{x}(k+1) = \boldsymbol{A}_e \boldsymbol{x}(k) + \boldsymbol{B}_e u(k) + f(k) \\
& s(k) = \boldsymbol{C}_e \boldsymbol{x}(k)
\end{aligned}
$$

其中 $\boldsymbol{A}_e = \boldsymbol{A}_2$，$\boldsymbol{B}_e = -\boldsymbol{B}_2$，$\boldsymbol{C}_e = [1, 0]$，$\boldsymbol{D}_e = 0$，$f(k) = \boldsymbol{R}_1 - \boldsymbol{A}_2 \boldsymbol{R}$。

未来的指令信号 $r(k+1)$ 未知，可采用线性外推的方法进行预测：

$$r(k+1) = 2r(k) - r(k-1)$$

取指令为正弦叠加信号 $r(k) = 1.0\sin(6\pi t) + 0.50\sin(10\pi t) + 0.30\sin(14\pi t)$，滑模参数

取 $c=30$。设被控对象的初值为 $\boldsymbol{x}=[1.5,0]$。在时间为 1s 处加入干扰信号 1.0，采用控制律式(3.36)，控制器输出在[−10,10]范围内。仿真结果如图 3-11～图 3-14 所示。

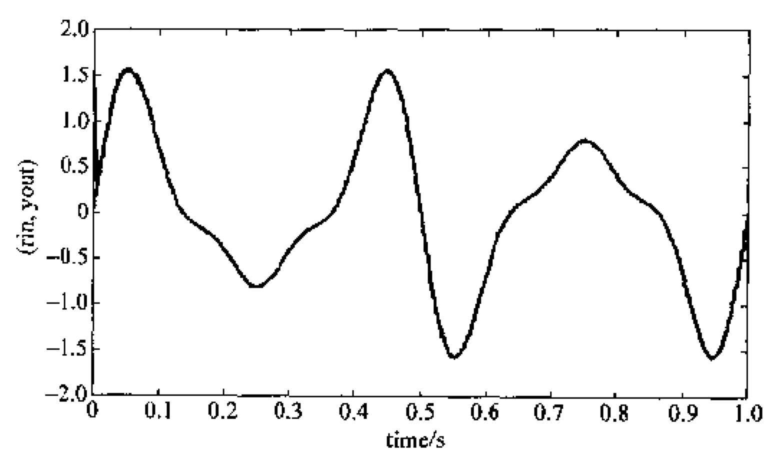


图 3-11 位置跟踪

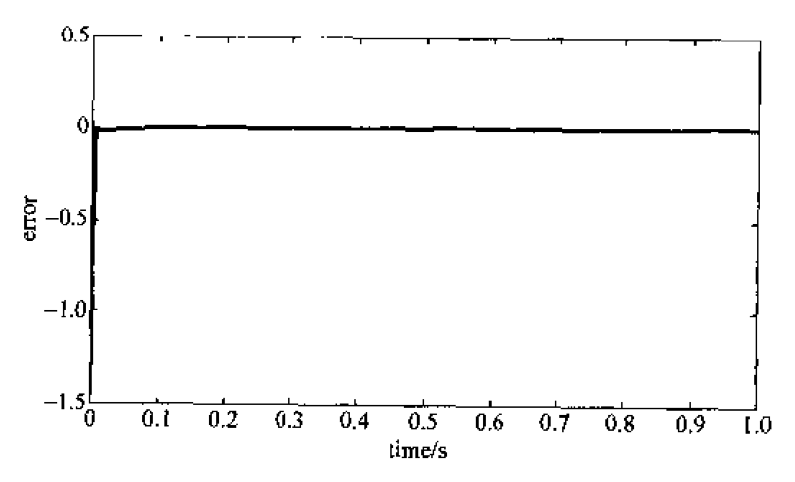


图 3-12 位置跟踪误差

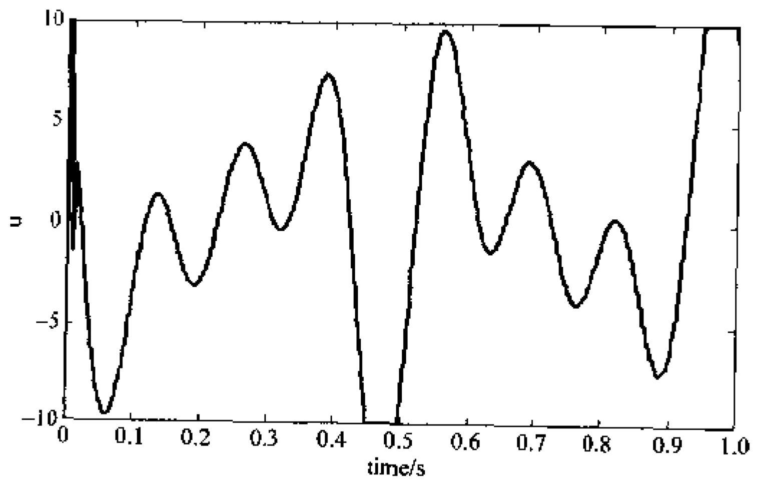


图 3-13 控制器输出

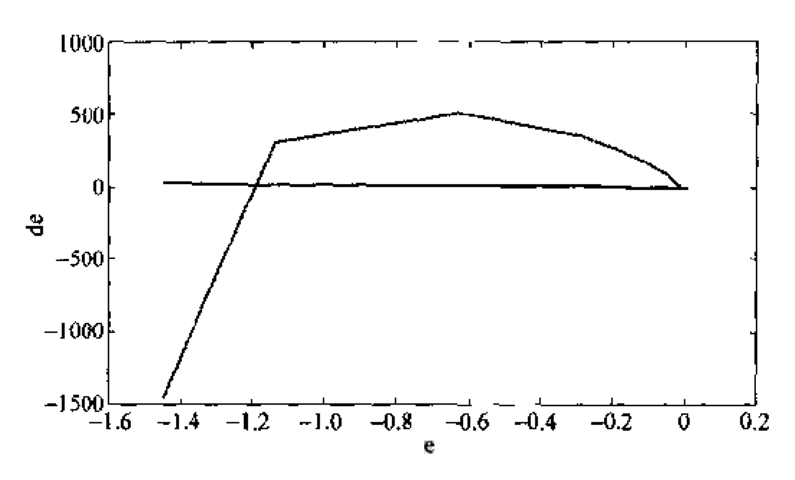


图 3-14　滑模运动相轨迹

仿真程序：chap3_3. m

```
clear all;
close all;

a = 25;b = 133;
ts = 0.001;
A1 = [0,1;0, - a];
B1 = [0;b];
C1 = [1,0];
D1 = 0;
[A2,B2,C2,D2] = c2dm(A1,B1,C1,D1,ts,'z');
Ae = A2;
Be = - B2;

r_1 = 0;r_2 = 0;
e_1 = 0;

x = [1.5;0];

c = 20;
Ce = [c,1];
for k = 1:1:1000
time(k) = k * ts;

r(k) = 1.0 * sin(2 * pi * 3 * k * ts) + 0.5 * sin(2 * pi * 5 * k * ts) + 0.3 * sin(2 * pi * 7 * k * ts);

% Using extrapolation method
   dr(k) = (r(k) - r_1)/ts;
   dr_1 = (r_1 - r_2)/ts;
   r1(k) = 2 * r(k) - r_1;
   dr1(k) = 2 * dr(k) - dr_1;
```

```
R = [r(k);dr(k)];
R1 = [r1(k);dr1(k)];
fk = R1 - A2 * R;

y(k) = x(1);

e(k) = r(k) - y(k);
 de(k) = (e(k) - e_1)/ts;
 s(k) = c * e(k) + de(k);

 f0 = 0.20 * 2 * abs(s(k))/(abs(Ce * Be) * (abs(e(k)) + abs(de(k))));
 deta1 = 0.5 * f0 * (Ce * Be) * (Ce * Be) * abs(e(k)) * (abs(e(k)) + abs(de(k)));
 deta2 = 0.5 * f0 * (Ce * Be) * (Ce * Be) * abs(de(k)) * (abs(e(k)) + abs(de(k)));

 cond1 = Ce * Be * s(k) * e(k);
if cond1< - deta1
   f1 = f0;
elseif abs(cond1)< = deta1
   f1 = 0;
elseif cond1>deta1
   f1 = - f0;
end

cond2 = Ce * Be * s(k) * de(k);
if cond2< - deta2
   f2 = f0;
elseif abs(cond2)< = deta2
   f2 = 0;
elseif cond2>deta2
   f2 = - f0;
end

Fd = [f1,f2];
ueq(k) = - inv(Ce * Be) * (Ce * (Ae - eye(2)) * [e(k);de(k)] + Ce * fk);
u(k) = ueq(k) + Fd * [e(k);de(k)];

x = A2 * x + B2 * u(k);

% Disturbance
if k == 1000
   u(k) = u(k) + 1.0;
end
if u(k)> = 10          % Restricting the output of controller
   u(k) = 10;
end
```

```
if u(k)<= -10
   u(k) = -10;
end

% Update Parameters
r_2 = r_1;r_1 = r(k);
e_1 = e(k);
end
figure(1);
plot(time,r,'r',time,y,'b');
xlabel('time(s)');ylabel('(rin,yout)');
figure(2);
plot(time,e,'r');
xlabel('time(s)');ylabel('error');
figure(3);
plot(time,u,'r');
xlabel('time(s)');ylabel('u');
figure(4);
plot(e,de,'r',e,-c*e,'b');
xlabel('e');ylabel('de');
```

3.5 基于趋近律的离散滑模控制位置跟踪

3.5.1 控制器设计

设二阶离散系统状态方程为

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}u(k) \tag{3.41}$$

其中 $\boldsymbol{x}(k)=[x_1(k),x_2(k)]$。

设位置指令为 $r(k)$，其变化率为 $\mathrm{d}r(k)$，取 $\boldsymbol{R}=[r(k);\mathrm{d}r(k)]$，$\boldsymbol{R}_1=[r(k+1);\mathrm{d}r(k+1)]$。采用线性外推的方法预测 $r(k+1)$ 及 $\mathrm{d}r(k+1)$，即

$$r(k+1)=2r(k)-r(k-1),\qquad \mathrm{d}r(k+1)=2\mathrm{d}r(k)-\mathrm{d}r(k-1) \tag{3.42}$$

切换函数为

$$s(k)=\boldsymbol{C}_\mathrm{e}E=\boldsymbol{C}_\mathrm{e}(\boldsymbol{R}-\boldsymbol{x}(k))$$

其中 $\boldsymbol{C}_\mathrm{e}=[c,1]$。则

$$\begin{aligned}s(k+1)&=\boldsymbol{C}_\mathrm{e}(\boldsymbol{R}_1-\boldsymbol{x}(k+1))\\&=\boldsymbol{C}_\mathrm{e}(\boldsymbol{R}_1-\boldsymbol{A}\boldsymbol{x}(k)-\boldsymbol{B}u(k))\\&=\boldsymbol{C}_\mathrm{e}\boldsymbol{R}_1-\boldsymbol{C}_\mathrm{e}\boldsymbol{A}\boldsymbol{x}(k)-\boldsymbol{C}_\mathrm{e}\boldsymbol{B}u(k)\end{aligned}$$

得到控制律为

$$u(k)=(\boldsymbol{C}_\mathrm{e}\boldsymbol{B})^{-1}(\boldsymbol{C}_\mathrm{e}\boldsymbol{R}_1-\boldsymbol{C}_\mathrm{e}\boldsymbol{A}\boldsymbol{x}(k)-s(k+1)) \tag{3.43}$$

基于指数趋近律的离散趋近律为

$$s(k+1)=s(k)+T(-\varepsilon\,\mathrm{sgn}(s(k))-qs(k)) \tag{3.44}$$

将式(3.44)代入式(3.43)，得到基于指数趋近律的离散控制律：

$$u(k) = (C_eB)^{-1}(C_eR_1 - C_eAx(k) - s(k) - \mathrm{d}s(k)) \tag{3.45}$$

其中 $\mathrm{d}s(k) = -\varepsilon T\mathrm{sgn}(s(k)) - qTs(k)$。

对基于趋近律的离散滑模变结构控制来说，有三个参数可调，即 q，ε 和 c。趋近速度参数 q 主要影响切换函数的动态过渡过程，适当调整该参数能够改变系统向滑模面的趋近速度，可以更好地改善系统动态品质。q 越大，系统到达滑模面的速度越快，尤其是 q 越接近于 $1/T$ 时，系统趋近速度最快。对于切换函数的设计，主要是设计滑模面参数 c(即滑模面斜率)，其目的是保证滑模运动渐近稳定且具有较快的动态响应速度。调节该参数对系统调节时间有较大的影响，滑模面参数 c 越大，滑模运动段响应越快，快速性越好。因而，增大 c 和 q 都可以相应提高系统的快速性。但是参数过大会导致控制量输出过大，并且在实际控制中，往往会引起系统的抖振。符号函数的增益参数 ε 是系统克服摄动及外干扰的主要参数，ε 越大，系统克服摄动和外干扰的能力就越强。但是，过大的增益将会导致系统抖振的加大，一般而言，系统的抖振幅度与 ε 成正比。

3.5.2 仿真实例

对象传递函数为

$$G(s) = \frac{133}{s(s+25)}$$

控制系统的采样周期为 0.001s，将系统离散化后的状态方程为

$$x(k+1) = Ax(k) + Bu(k)$$

其中 $A = \begin{bmatrix} 1 & 0.0010 \\ 0 & 0.9753 \end{bmatrix}$，$B = \begin{bmatrix} 0.0001 \\ 0.1314 \end{bmatrix}$。

指令取正弦信号 $r(k) = 0.5\sin(6\pi t)$，其中频率 $F = 3\mathrm{Hz}$，控制器参数为 $c=10$，$\varepsilon=5$，$q=30$。初始状态取$[-0.5, -0.5]$。

分别采用 M 语言和 Simulink 环境进行仿真。

1. 仿真方法之一：采用 M 语言，仿真结果如图 3-15～图 3-17 所示

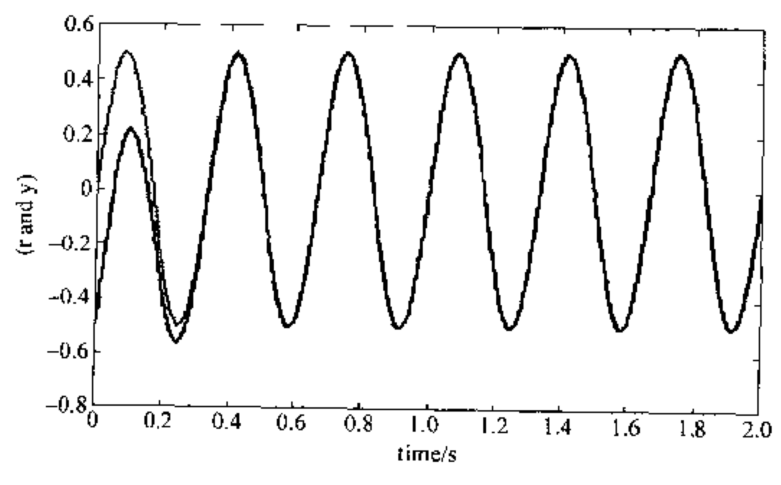


图 3-15 正弦信号跟踪

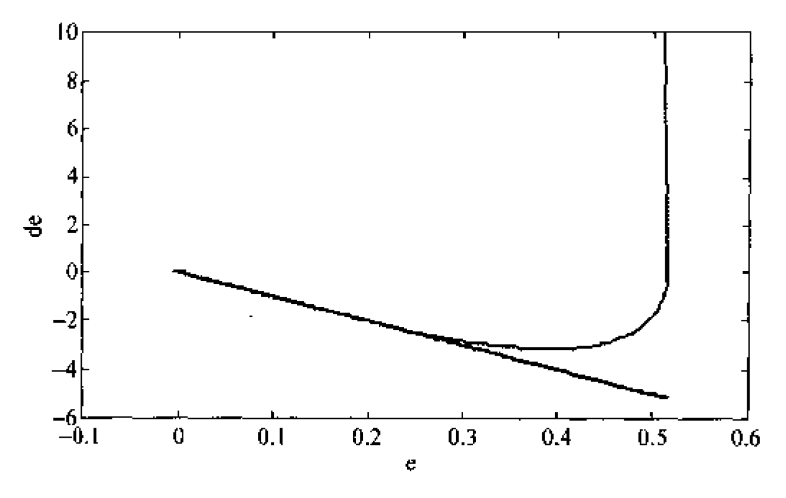


图 3-16　相轨迹

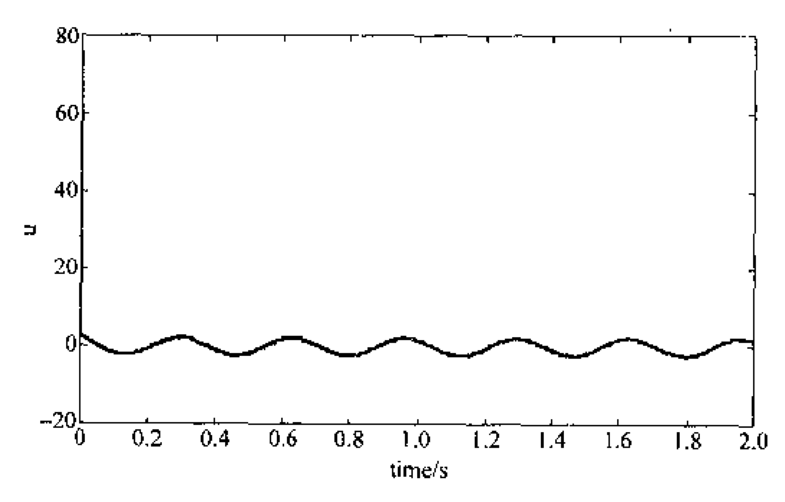


图 3-17　控制器输出

仿真程序：chap3_4. m

```
% Discrete Reaching Law VSS Control
clear all;
close all;

a = 25;b = 133;

ts = 0.001;
A1 = [0,1;0, - a];
B1 = [0;b];
C1 = [1,0];
D1 = 0;
[A,B,C,D] = c2dm(A1,B1,C1,D1,ts,'z');

x = [ - 0.5; - 0.5];
r_1 = 0;r_2 = 0;

for k = 1:1:2000
```

```
time(k) = k * ts;

    r(k) = 0.5 * sin(3 * 2 * pi * k * ts);
    c = 10;eq = 5;q = 30;
    Ce = [c,1];

    % Using Waitui method
    dr(k) = (r(k) - r_1)/ts;
    dr_1 = (r_1 - r_2)/ts;
    r1(k) = 2 * r(k) - r_1;
    dr1(k) = 2 * dr(k) - dr_1;

    R = [r(k);dr(k)];
    R1 = [r1(k);dr1(k)];

    E = R - x;
    e(k) = E(1);
    de(k) = E(2);

    s(k) = Ce * E;

    ds(k) = - eq * ts * sign(s(k)) - q * ts * s(k);
     u(k) = inv(Ce * B) * (Ce * R1 - Ce * A * x - s(k) - ds(k));

     x = A * x + B * u(k);
     y(k) = x(1);

% Update Parameters
r_2 = r_1;
r_1 = r(k);
end
figure(1)
plot(time,r,'r',time,y,'b');
xlabel('Time(second)');ylabel('(r and y)');
figure(2)
plot(time,s,'r');
xlabel('Time(second)');ylabel('Switch function s');
figure(3)
plot(e,de,'r',e, - c * e,'b');
xlabel('e');ylabel('de');
figure(4)
plot(time,u,'r');
xlabel('Time(second)');ylabel('u');
```

2. 仿真方法之二：采用 Simulink 环境

利用 S 函数实现离散控制器的设计及仿真结果的输出。在 S 函数中，采用初始化、更新函数和输出函数，即 mdlInitializeSizes 函数、mdlUpdates 函数和 mdlOutputs 函数。在 S 函数的初始化函数中采用 sizes 结构，选择 1 个输出，5 个输入。其中前 3 个输入为位置指

令及其延迟，后 2 个输入为实际位置及其延迟。S 函数嵌入在 Simulink 程序中。仿真结果如图 3-18～图 3-20 所示。

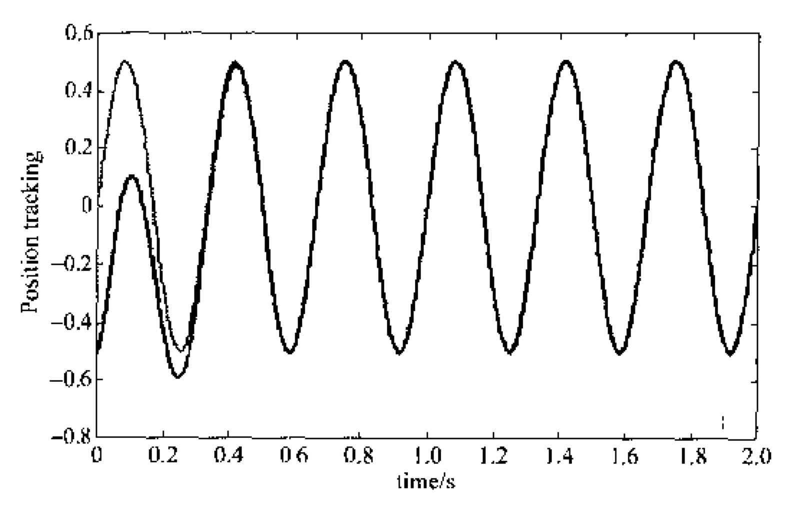


图 3-18　正弦信号跟踪

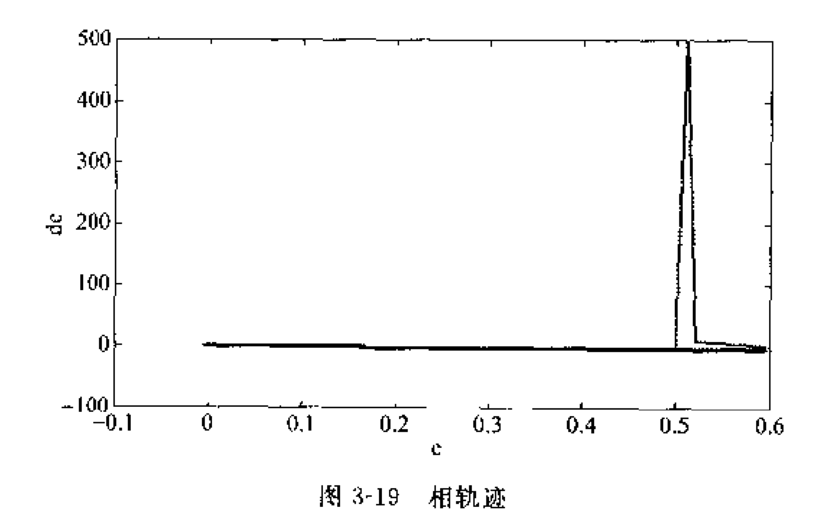


图 3-19　相轨迹

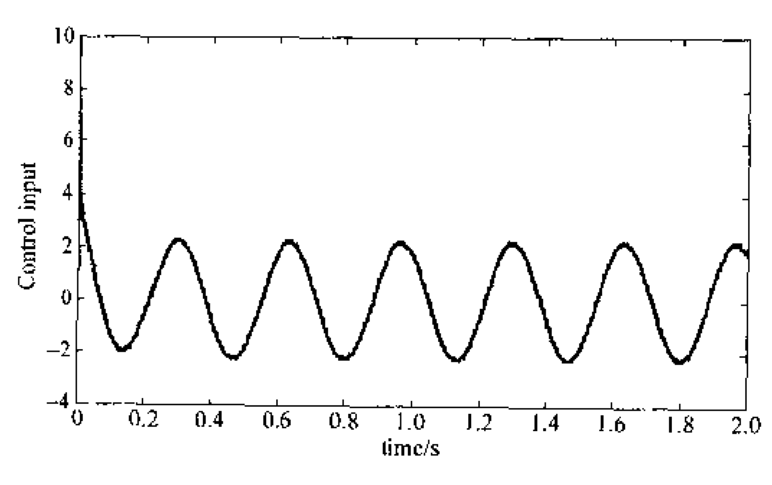


图 3-20　控制器输出

仿真程序如下。

(1) 初始化程序：chap3_5int.m

```
clear all;
close all;

a = 25;b = 133;

T = 0.001;
A1 = [0,1;0,-a];
B1 = [0;b];
C1 = [1,0];
D1 = 0;
[A,B,C,D] = c2dm(A1,B1,C1,D1,T,'z');
```

(2) Simulink 主程序(如图 3-21 所示)：chap3_5sim.mdl

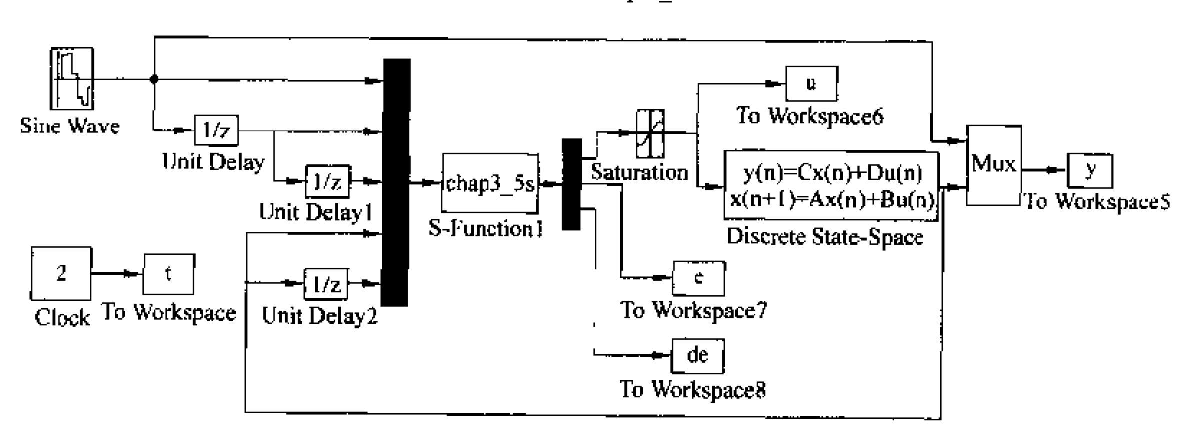


图 3-21 主程序图

(3) 控制器 S 函数：chap3_5s.m

```
function [sys,x0,str,ts] = exp_pidf(t,x,u,flag)
switch flag,
case 0              % initializations
    [sys,x0,str,ts] = mdlInitializeSizes;
case 2              % discrete states updates
    sys = mdlUpdates(x,u);
case 3              % computation of control signal
%     sys = mdlOutputs(t,x,u,kp,ki,kd,MTab);
    sys = mdlOutputs(t,x,u);
case {1, 4, 9}      % unused flag values
    sys = [];
otherwise           % error handling
    error(['Unhandled flag = ',num2str(flag)]);
end;

%==============================================
% when flag = 0, perform system initialization
%==============================================
```

```
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;           % read default control variables
sizes.NumContStates = 0;    % no continuous states
sizes.NumDiscStates = 3;    % 3 states
sizes.NumOutputs = 3;       % 1 output variables: control u(t) and state x(3)
sizes.NumInputs = 5;        % 5 input signals
sizes.DirFeedthrough = 1;   % input reflected directly in output
sizes.NumSampleTimes = 1;   % single sampling period
sys = simsizes(sizes);      %
x0 = [0; 0; 0];             % zero initial states
str = [];
ts = [-1 0];                % sampling period
% ==========================================================
% when flag = 2, updates the discrete states
% ==========================================================
function sys = mdlUpdates(x,u)
  T = 0.001;

sys = [(u(1) - u(2))/T;
    (u(2) - u(3))/T;
    (u(4) - u(5))/T];

% ==========================================================
% when flag = 3, computates the output signals
% ==========================================================
function sys = mdlOutputs(t,x,u,kp,ki,kd,MTab)
  T = 0.001;

  r = u(1);
  r_1 = u(2);
  r_2 = u(3);
  dr = x(1);
  dr_1 = x(2);

  xp(1) = u(4);
  xp(2) = x(3);

  c = 10;eq = 5;q = 30;
  Ce = [c,1];


A = [1.0000    0.0010;
  0     0.9753];
B = [0.0001;
    0.1314];

  % Using Waitui method
  r1 = 2 * r - r_1;
```

```
    dr1 = 2 * dr - dr_1;

    R = [r;dr];
    R1 = [r1;dr1];

    E = R - xp';
    e = E(1);
    de = E(2);

    s = Ce * E;

    ds = -eq * T * sign(s) - q * T * s;
    ut = inv(Ce * B) * (Ce * R1 - Ce * A * xp' - s - ds);

sys(1) = ut;
sys(2) = e;
sys(3) = de;
```

(4) 作图程序：chap3_5plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('position tracking');

figure(2);
plot(t,u,'r');
xlabel('time(s)');ylabel('control input');

figure(3)
c = 10;
plot(e,de,'r',e,-c*e,'b');
xlabel('e');ylabel('de');
```

3.6 基于滤波器的趋近律滑模控制

在 3.5 节的基础上，在控制系统中加入随机干扰和随机噪声，采用离散 Kalman 滤波器对干扰和噪声进行滤波。

3.6.1 Kalman 滤波器的设计

针对以下离散线性系统

$$\left.\begin{aligned} \boldsymbol{x}(k) &= \boldsymbol{A}\boldsymbol{x}(k-1) + \boldsymbol{B}(\boldsymbol{u}(k) + \boldsymbol{w}(k)) \\ \boldsymbol{y}_v(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{v}(k) \end{aligned}\right\} \tag{3.46}$$

其中，$\boldsymbol{w}(k)$ 为过程噪声，$\boldsymbol{v}(k)$ 为测量噪声，$\boldsymbol{y}_v(k)$ 为被控对象实际输出。

控制方法采用基于趋近律的滑动模态控制，即控制器采用式(3.45)。

离散 Kalman 滤波器算法如下：

$$M_n(k)=\frac{P(k)C^T}{CP(k)C^T+R} \tag{3.47}$$

$$P(k)=AP(k-1)A^T+BQB^T \tag{3.48}$$

$$P(k)=(I_n-M_n(k)C)P(k) \tag{3.49}$$

$$x(k)=Ax(k-1)+M_n(k)(y_v(k)-CAx(k-1)) \tag{3.50}$$

滤波后被控对象的输出为：

$$y_e(k)=Cx(k) \tag{3.51}$$

带有 Kalman 滤波器的控制系统结构如图 3-22 所示。

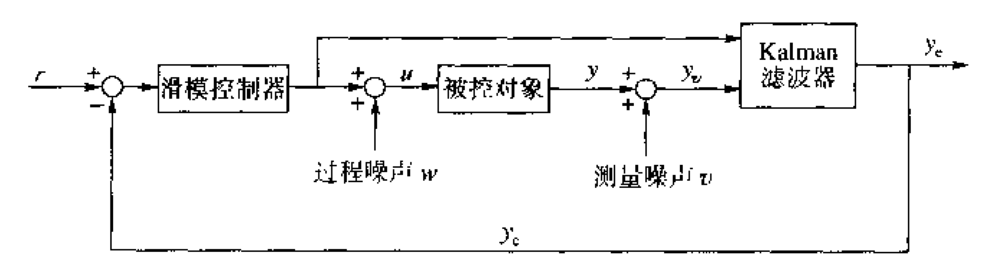


图 3-22　控制系统结构

3.6.2 仿真实例

对象传递函数为

$$G(s)=\frac{133}{s(s+25)}$$

取采样时间为 1ms，被控对象可转化为以下离散线性系统的形式：

$$\begin{cases}x(k)=Ax(k-1)+B(u(k)+w(k))\\ y_v(k)=Cx(k)+v(k)\end{cases}$$

其中 $A=\begin{bmatrix}1 & 0.0010\\ 0 & 0.9753\end{bmatrix}$，$B=\begin{bmatrix}0.0001\\ 0.1314\end{bmatrix}$，$C=[1,0]$，$D=0$，$w(k)$为$[-0.5,0.5]$内的白噪声信号，$v(k)$为$[-0.2,0.2]$的白噪声信号。

在 Kalman 滤波算法中，取 $Q=1$，$R=1$。

位置输入指令为

$$r(k)=A\sin(2\pi Ft)$$

其中 $t=k\times T$，取 $k=1000$，$A=0.50$，$F=1.0$。

控制律为式(3.45)，取控制器参数为 $c=30$，$\varepsilon=150$，$q=300$。当 $M=2$ 时，不采用 Kalman 滤波算法，位置跟踪及相轨迹如图 3-23 和图 3-24 所示。当 $M=1$ 时，采用 Kalman 滤波算法，位置跟踪及相轨迹如图 3-25～图 3-27 所示。由仿真结果可见，采用 Kalman 滤波器后，控制精度大大提高。

仿真程序：chap3_6.m

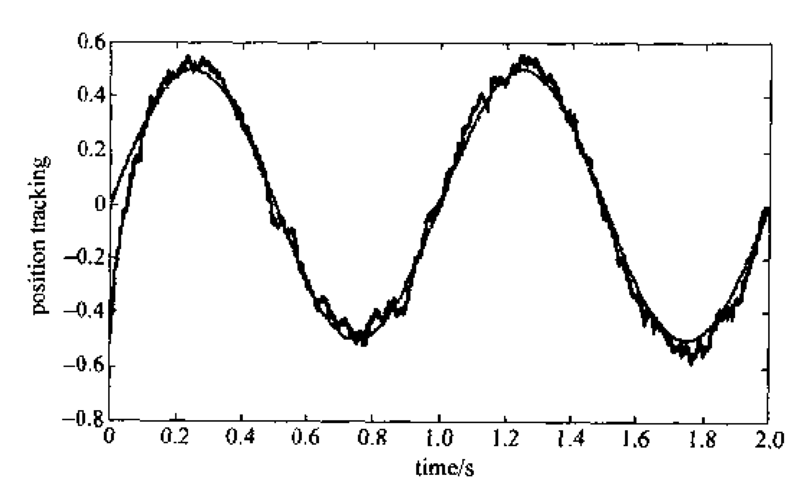


图 3-23 未加滤波器时的位置跟踪（$M=2$）

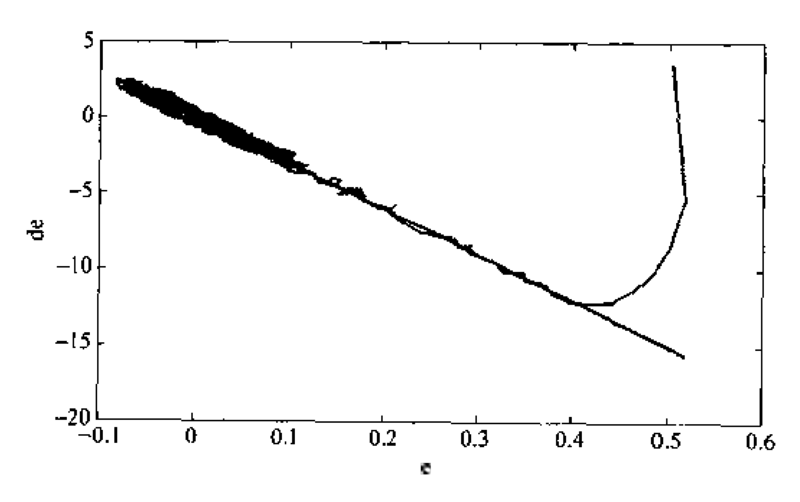


图 3-24 未加滤波器时的相轨迹（$M=2$）

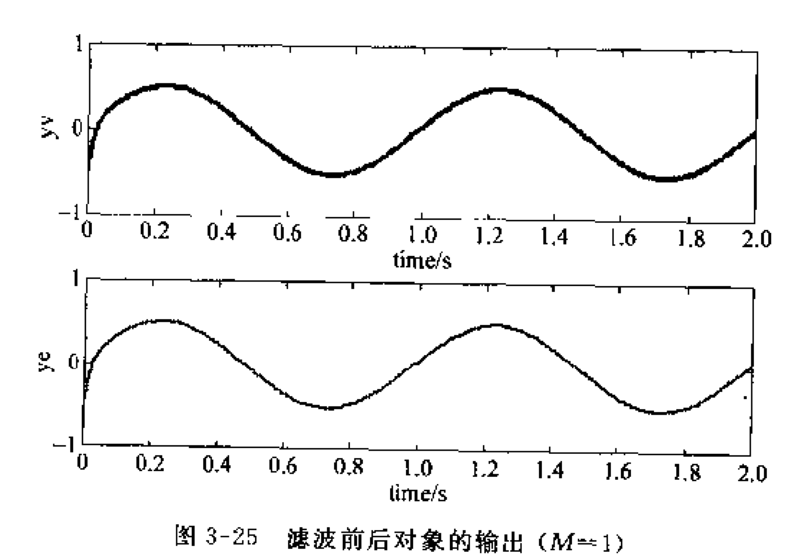


图 3-25 滤波前后对象的输出（$M=1$）

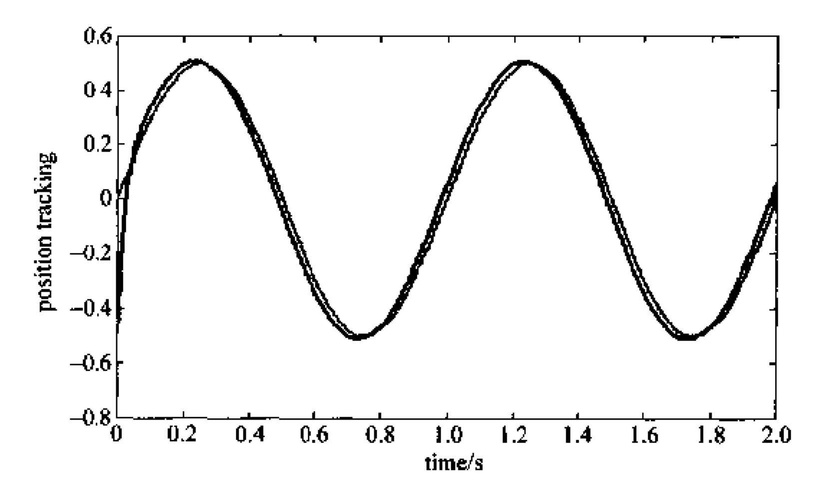


图 3-26　加滤波器后的位置跟踪（$M=1$）

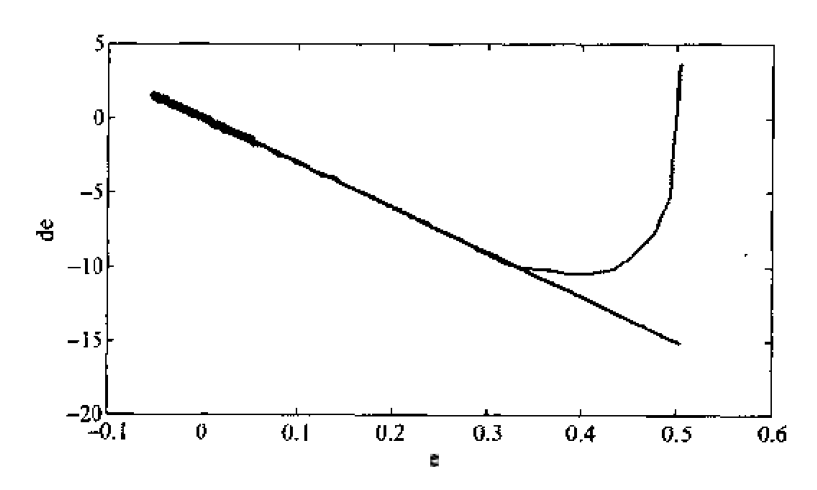


图 3-27　加入滤波器后的相轨迹（$M=1$）

```
% Discrete Reaching Law VSS Control based on Kalman Filter
clear all;
close all;

a = 25;b = 133;

ts = 0.001;
A1 = [0,1;0, - a];
B1 = [0;b];
C1 = [1,0];
D1 = 0;
[A,B,C,D] = c2dm(A1,B1,C1,D1,ts,'z');

x = [ - 0.5; - 0.5];
r_1 = 0;r_2 = 0;

Q = 10;                 % Covariances of w
Rm = 10;                % Covariances of v
P = B * Q * B';         % Initial error covariance
```

```
for k = 1:1:2000
    time(k) = k * ts;

    r(k) = 0.5 * sin(1 * 2 * pi * k * ts);
    c = 30;eq = 150;q = 300;
    Ce = [c,1];

 % Using Waitui method
    dr(k) = (r(k) - r_1)/ts;
    dr_1 = (r_1 - r_2)/ts;
    r1(k) = 2 * r(k) - r_1;
    dr1(k) = 2 * dr(k) - dr_1;

    R = [r(k);dr(k)];
    R1 = [r1(k);dr1(k)];

    E = R - x;
    e(k) = E(1);
    de(k) = E(2);

    s(k) = Ce * E;
    ds(k) = - eq * ts * sign(s(k)) - q * ts * s(k);

    u(k) = inv(Ce * B) * (Ce * R1 - Ce * A * x - s(k) - ds(k));
     wn(k) = rands(1);          % Process noise on u
     u(k) = u(k) + wn(k);

    x = A * x + B * u(k);
    v(k) = 0.015 * rands(1);   % Measurement noise on y
    yv(k) = C * x + v(k);

 M = 1;
 if M == 1              % Kalman Filter
     Mn = P * C'/(C * P * C' + Rm);
     P = A * P * A' + B * Q * B';
     P = (eye(2) - Mn * C) * P;
     x = A * x + Mn * (yv(k) - C * A * x);
     ye(k) = C * x;
 elseif M == 2       % No Filter
       ye(k) = yv(k);
     x(1) = ye(k);
 end


% Update Parameters
r_2 = r_1;
r_1 = r(k);
end

figure(1);
subplot(211);
```

```
plot(time,yv,'b');
xlabel('Time(s)');ylabel('yv');
subplot(212);
plot(time,ye,'r');
xlabel('Time(s)');ylabel('ye');

figure(2);
plot(time,r,'r',time,ye,'b');
xlabel('Time(second)');ylabel('position tracking');
figure(3);
plot(time,s,'r');
xlabel('Time(second)');ylabel('Switch function s');
figure(4);
plot(e,de,'r',e,-c*e,'b');
xlabel('e');ylabel('de');
figure(5);
plot(time,u,'r');
xlabel('Time(second)');ylabel('u');
```

3.7　基于变速趋近律的滑模控制

指数趋近律的离散形式有它自身的缺点，即切换带为带状，当系统在切换带中运动时，最后不能趋近于原点，而是趋近于原点附近的一个抖振。这种抖振将可能激励系统中存在的未建模高频成分，并可能增加控制器的负担。

3.7.1　变速趋近律及控制

文献[3]研究了连续变结构控制系统的变速趋近律：

$$\dot{s}=-\varepsilon\|\boldsymbol{x}\|_1\operatorname{sgn}(s) \tag{3.52}$$

其中 $\|\boldsymbol{x}\|_1=\sum_{i=1}^{n}|x_i|$ 为系统状态范数。

变速趋近律采用全部状态变量构成的控制，趋近速度为 $\varepsilon\|\boldsymbol{x}\|_1$，与 $\|\boldsymbol{x}\|_1$ 成比例，比例系数是 ε。如果 $\|\boldsymbol{x}\|_1$ 很大，且 ε 较大，则到达切换面时，系统将具有较大的速度，会引起较大的抖振；如果 ε 太小，则趋近速度很慢，正常运动段将是慢速的，调节时间长。由于 $\dot{s}s=-\varepsilon\|\boldsymbol{x}\|_1\operatorname{sgn}(s)s=-\varepsilon\|\boldsymbol{x}\|_1|s|<0$，因而，变速趋近律满足滑动模态的存在性和到达性条件，所以可以将系统引导到滑动模态上。

将此变速趋近律应用到离散系统中，其相应的离散形式为

$$s(k+1)-s(k)=-\varepsilon T\|\boldsymbol{x}(k)\|_1\operatorname{sgn}(s(k)) \tag{3.53}$$

考虑二阶离散系统状态方程如下：

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}u(k)\qquad \boldsymbol{x}\in\mathbf{R}^n,\ u\in\mathbf{R} \tag{3.54}$$

设位置指令为 $r(k)$，其变化率为 $dr(k)$，则 $\boldsymbol{R}=[r(k),dr(k)]$，$\boldsymbol{R}_1=[r(k+1),dr(k+1)]$。$r(k+1)$ 及 $dr(k+1)$ 采用线性外推的方法进行预测：

$$r(k+1) = 2r(k) - r(k-1), \qquad \mathrm{d}r(k+1) = 2\mathrm{d}r(k) - \mathrm{d}r(k-1)$$

切换函数为

$$s(k) = \boldsymbol{C}_e E = \boldsymbol{C}_e(\boldsymbol{R}(k) - \boldsymbol{x}(k)) \tag{3.55}$$

其中 $\boldsymbol{C}_e = [c, 1]$。则

$$\begin{aligned} s(k+1) &= \boldsymbol{C}_e(\boldsymbol{R}(k+1) - \boldsymbol{x}(k-1)) \\ &= \boldsymbol{C}_e(\boldsymbol{R}(k+1) \quad \boldsymbol{A}\boldsymbol{x}(k) - \boldsymbol{B}u(k)) \\ &= \boldsymbol{C}_e\boldsymbol{R}(k+1) - \boldsymbol{C}_e\boldsymbol{A}\boldsymbol{x}(k) - \boldsymbol{C}_e\boldsymbol{B}u(k) \end{aligned}$$

得到控制律为

$$u(k) = (\boldsymbol{C}_e\boldsymbol{B})^{-1}(\boldsymbol{C}_e\boldsymbol{R}(k+1) - \boldsymbol{C}_e\boldsymbol{A}\boldsymbol{x}(k) - s(k+1)) \tag{3.56}$$

将式(3.53)代入式(3.56),得到基于变速趋近律的离散控制律:

$$u(k) = (\boldsymbol{C}_e\boldsymbol{B})^{-1}(\boldsymbol{C}_e\boldsymbol{R}(k+1) - \boldsymbol{C}_e\boldsymbol{A}\boldsymbol{x}(k) - s(k) - \mathrm{d}s(k)) \tag{3.57}$$

其中 $\mathrm{d}s(k) = -\varepsilon T \| \boldsymbol{x}(k) \|_1 \mathrm{sgn}(s(k))$。

设采样时间 T 很小时,离散滑动模态控制律式(3.57)满足离散滑模的存在性和到达性条件[4]:

$$(s(k+1) - s(k))\mathrm{sgn}(s(k)) < 0, \qquad (s(k+1) + s(k))\mathrm{sgn}(s(k)) > 0 \tag{3.58}$$

稳定性证明:

由于

$$\begin{aligned} (s(k+1) - s(k))\mathrm{sgn}(s(k)) &= (-\varepsilon T \| \boldsymbol{x}(k) \|_1 \mathrm{sgn}(s(k)))\mathrm{sgn}(s(k)) \\ &= -\varepsilon T \| \boldsymbol{x}(k) \|_1 \mid s(k) \mid < 0 \end{aligned} \tag{3.59}$$

当采样时间 T 很小时,$2c - \varepsilon T > 0$,$2 - \varepsilon T > 0$,有

$$\begin{aligned} (s(k+1) + s(k))\mathrm{sgn}(s(k)) &= (2s(k) - \varepsilon T \| \boldsymbol{x}(k) \|_1 \mathrm{sgn}(s(k)))\mathrm{sgn}(s(k)) \\ &= 2 \mid s(k) \mid - \varepsilon T \| \boldsymbol{x}(k) \|_1 \\ &\leqslant 2c \mid x_1 \mid + 2 \mid x_2 \mid - \varepsilon T(\mid x_1 \mid + \mid x_2 \mid) \\ &= (2c - \varepsilon T) \mid x_1 \mid + (2 - \varepsilon T) \mid x_2 \mid > 0 \end{aligned} \tag{3.60}$$

可见,滑模趋近律式(3.53)满足离散滑动模态的存在性和到达性条件,所设计的控制系统是稳定的。

离散滑模控制系统中的运动从任意初始点出发,单调地向滑模面运动,在有限时间内到达滑模面。一旦穿越滑模面以后,其每一个后继步都从另一面穿越滑模面,并一直进行下去。其每一步的长度非递增,其运动轨迹限于一特定带内,这个特定带就是切换区,定义切换区为

$$\{x \in \mathbf{R}^n \mid -\Delta < s(k) < \Delta\} \tag{3.61}$$

对于指数趋近律,由式(3.44)可知,当 $s(k) = 0^+$ 时,$s(k+1) = -\varepsilon T$;当 $s(k) = 0^-$ 时,$s(k+1) = \varepsilon T$。说明指数趋近律的切换带是不过原点的宽度为 $2\varepsilon T$ 的带状,表明稳态时,系统滑模函数在这两个值之间来回切换。可见如果不考虑其他造成抖振的因素,符号函数的增益值大小直接决定了系统稳态时的抖振幅度。为了减小抖振现象,可尽可能地减小符号函数的增益。

对于变速趋近律,由式(3.53)得

$$\text{当 } s(k) = 0^+ \text{ 时}, s(k+1) = -\varepsilon T \| \boldsymbol{x}(k) \|_1 \tag{3.62}$$

$$\text{当 } s(k) = 0^- \text{ 时}, s(k+1) = \varepsilon T \| \boldsymbol{x}(k) \|_1 \tag{3.63}$$

上两式说明在二阶系统中，变速趋近律的切换带是由经过原点的两条射线组成，并且把 $s=0$ 夹在其中，切换带带宽为

$$2\Delta = 2\varepsilon T \| x(k) \|_1 \tag{3.64}$$

由式(3.64)可见，采样周期大时系统的抖振比采样周期小时的抖振大，运动进入切换带后，穿越切换面的幅度将与 $\| x(k) \|_1$ 成比例。因此当系统稳态后，可稳定于原点，具有良好的稳态性能。但是在系统刚进入切换带时，由于 $\| x(k) \|_1$ 比较大，会产生大幅度的抖振，这是变速趋近律的不足之处。

图3-28和图3-29分别表示变速趋近律和指数趋近律的相轨迹，前者趋近于原点，后者趋近于原点附近的一个抖振；前者的切换带为扇形，后者的切换带为带状。图中切换区用虚线表示。

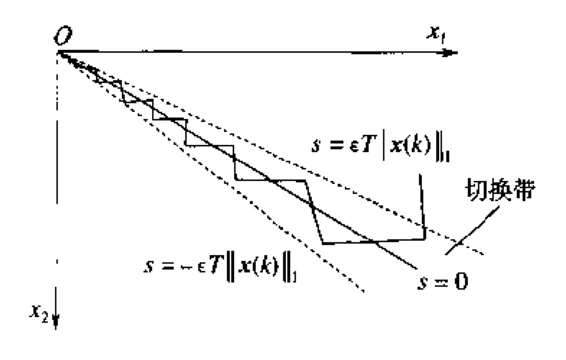


图3-28　变速趋近律相轨迹

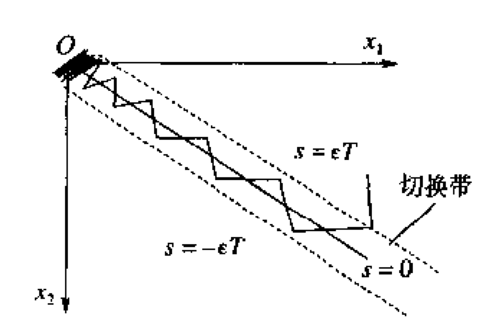


图3-29　指数趋近律相轨迹

3.7.2　基于组合趋近律的控制

指数趋近律的切换带是不过原点的宽度为 $2\varepsilon T$ 的带状，在稳态时，滑模函数在这两个值之间来回切换，会产生较大的稳态抖振。采用变速趋近律，切换带是由经过原点的两条射线组成，并把 $s=0$ 夹在其中，稳态时，可稳定于原点，有效地降低了稳态时的抖振，具有良好的稳态性能。但当系统刚进入切换带时，由于 $\| x(k) \|_1$ 比较大，变速趋近律会产生大幅度的抖振。

如果把指数趋近律和变速趋近律结合起来，即在滑模运动的前期，采用基于指数趋近律的控制律式(3.45)，在滑模运动的后期和稳定段，采用基于变速趋近律的控制律式(3.57)，可以克服两种趋近律的缺点，保留它们的优点，从而使系统性能达到最好。即选定一个正的实数 k_0，当 $\| x(k) \|_1 > k_0$ 时，采用指数趋近律；当 $\| x(k) \|_1 \leqslant k_0$ 时，采用变速趋近律。k_0 的选定必须适当，选得太大，会掩盖变速趋近律的优点；选得太小，则可能产生大幅度穿越抖振。两种控制律转折点的选择可根据实际情况而定。

由式(3.44)和式(3.53)可得到系统的组合控制律：

$$u(k) = \begin{cases} (3.45) & \| x(k) \|_1 > k_0 \\ (3.57) & \| x(k) \|_1 \leqslant k_0 \end{cases} \tag{3.65}$$

3.7.3 仿真实例

对象传递函数为

$$G(s)=\frac{133}{s(s+25)}$$

控制系统的采样周期为 0.001s，将系统离散化后的状态方程为

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}u(k)$$

其中 $\boldsymbol{A}=\begin{bmatrix}1 & 0.0010\\0 & 0.9753\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0.0001\\0.1314\end{bmatrix}$。

在组合趋近律中，指令取阶跃信号 $r(k)=10$，控制器参数为 $c=20$；$\varepsilon=5$，$q=30$，按下式确定控制律的转换点：

$$\|\boldsymbol{x}(k)\|_1=0.60$$

被控对象初值取[−0.8，−0.5]，控制输入信号范围为[−10，+10]。分别采用指数趋近律、变速趋近律和组合控制律。取 $M=1$ 时，采用指数趋近律，仿真结果如图 3-30～图 3-32 所示。取 $M=2$ 时，采用变速趋近律，仿真结果如图 3-33～图 3-35 所示。取 $M=3$ 时，采用组合控制律，仿真结果如图 3-36～图 3-38 所示。

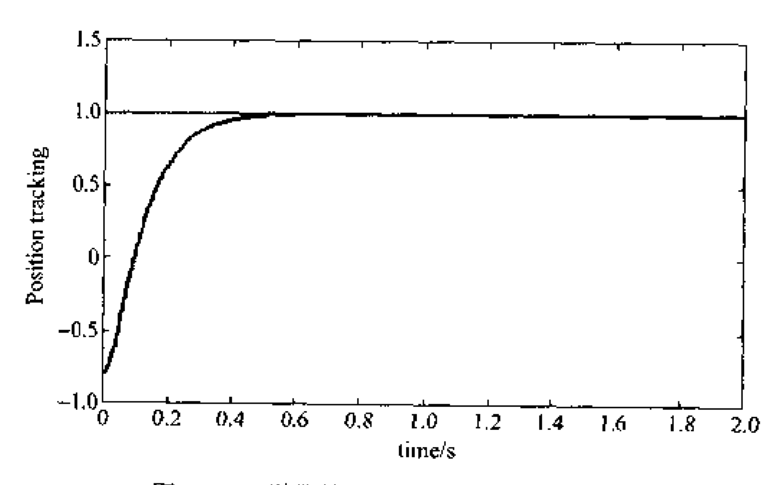


图 3-30　指数趋近律的阶跃响应（$M=1$）

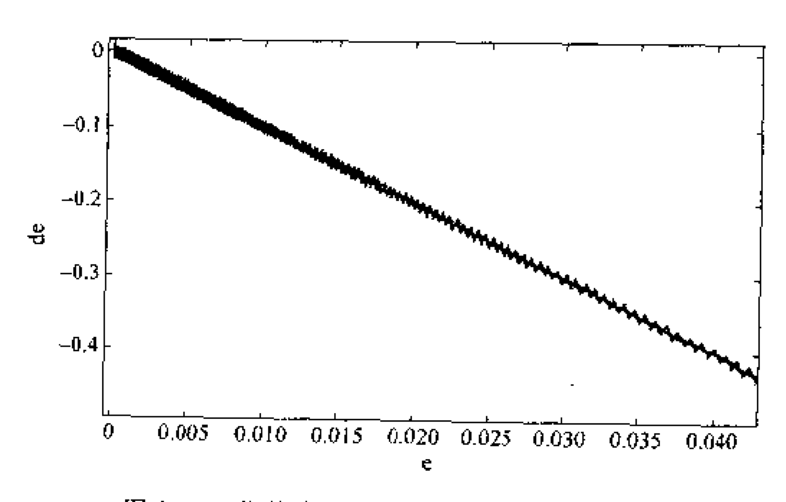


图 3-31　指数趋近律的相轨迹放大图（$M=1$）

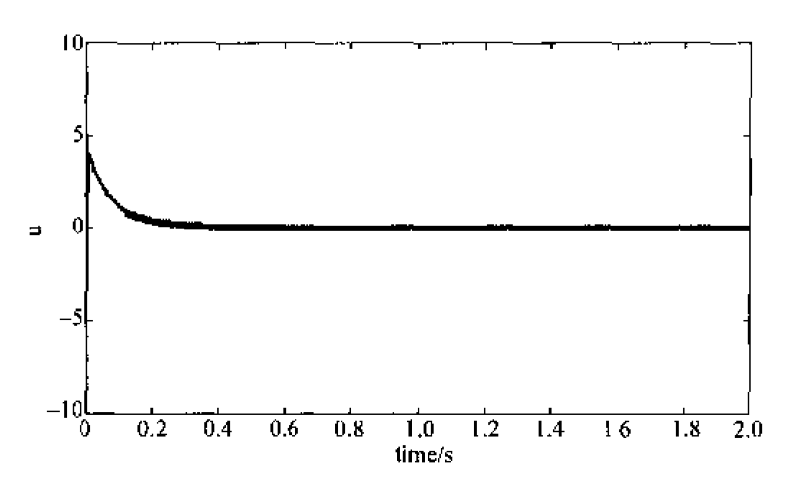


图 3-32　指数趋近律的控制信号($M=1$)

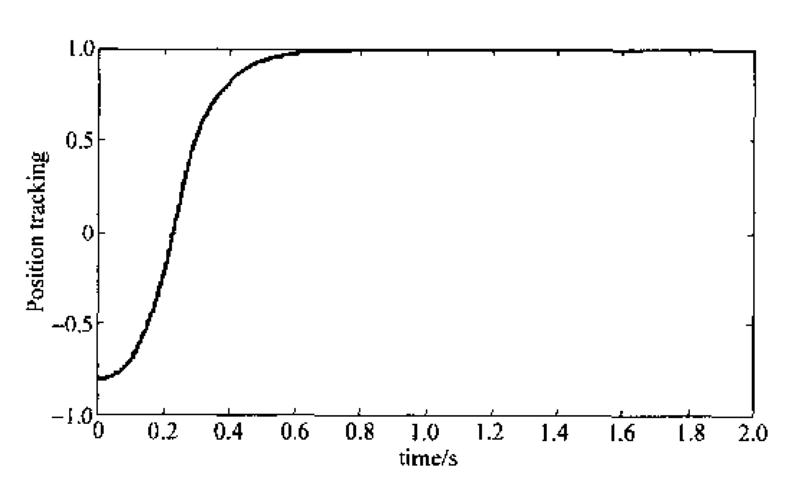


图 3-33　变速趋近律的阶跃响应 ($M=2$)

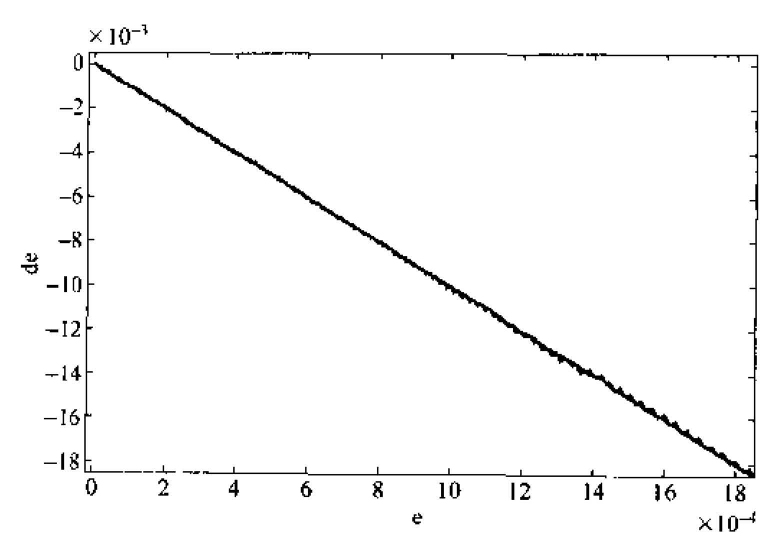


图 3-34　变速趋近律控制的相轨迹放大图 ($M=2$)

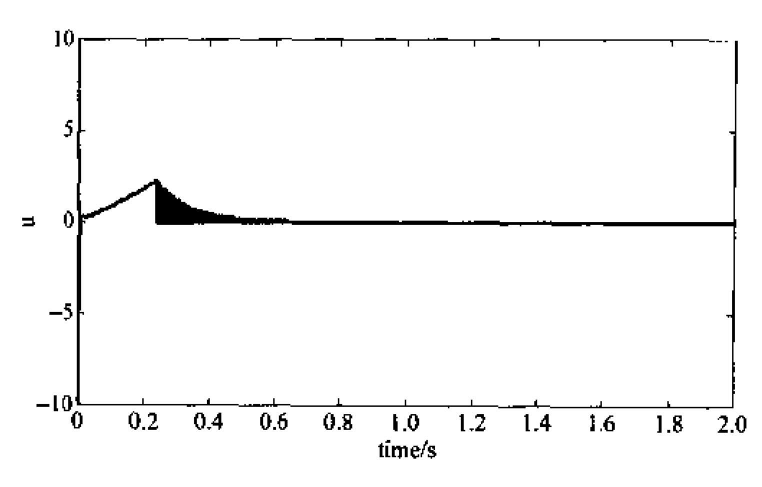


图 3-35 变速趋近律的控制信号（$M=2$）

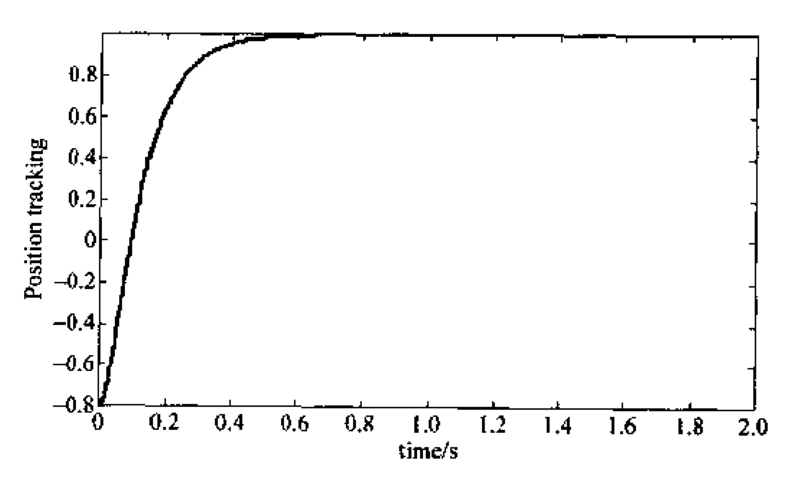


图 3-36 组合趋近律的阶跃响应（$M=3$）

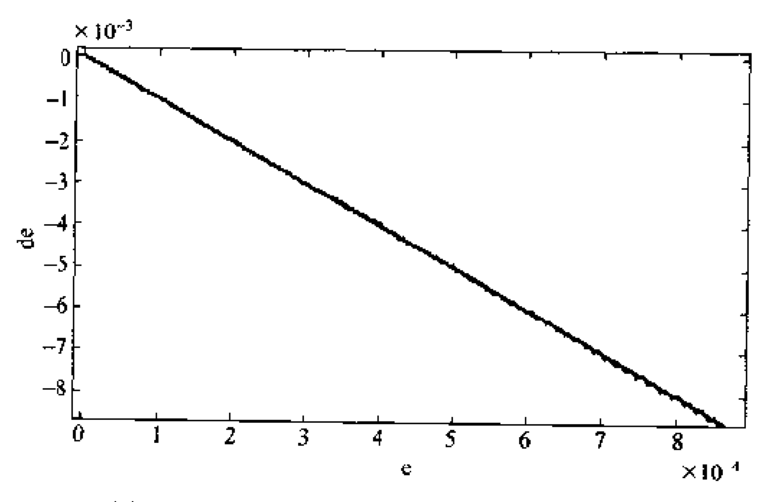


图 3-37 组合趋近律的相轨迹放大图（$M=3$）

由仿真结果可见，指数趋近律控制时的切换带为带状，当系统在切换带中运动时，最后不能趋近于原点，而是趋近于原点附近的一个抖振。变速趋近律控制可以使系统进入稳态后趋近原点，但在刚进入切换带时，有大幅度的抖振。组合控制结合两种方法的优点，克服

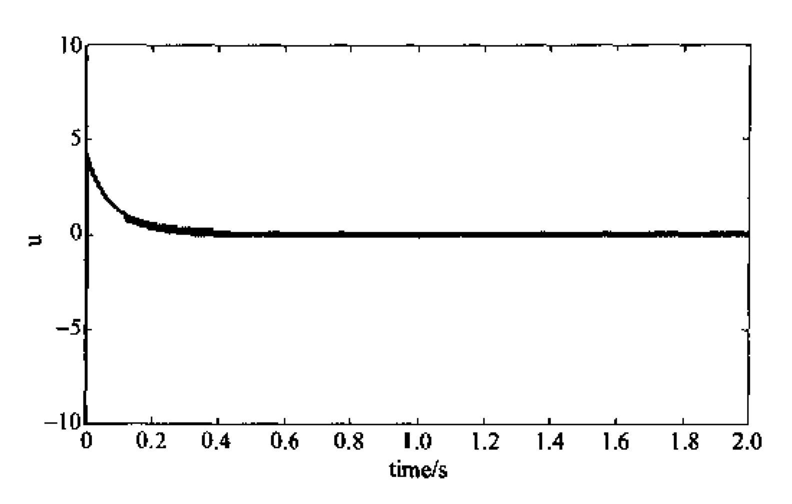


图 3-38　组合趋近律的控制信号（$M=3$）

它们各自的缺点，能达到很好的控制效果。

仿真程序：chap3_7.m

```
clear all;
close all;

a = 25;b = 133;
ts = 0.001;
A1 = [0,1;0, - a];
B1 = [0;b];
C1 = [1,0];
D1 = 0;
[A,B,C,D] = c2dm(A1,B1,C1,D1,ts,'z');

x = [ - 0.8; - 0.5];
r_1 = 0;r_2 = 0;

c = 20;
eq = 5;
q = 30;
Ce = [c,1];
for k = 1:1:2000
    time(k) = k * ts;
   r(k) = 1.0;

   % Using Waitui method
   dr(k) = (r(k) - r_1)/ts;
   dr_1 = (r_1 - r_2)/ts;
   r1(k) = 2 * r(k) - r_1;
   dr1(k) = 2 * dr(k) - dr_1;

   R = [r(k);dr(k)];
   R1 = [r1(k);dr1(k)];
```

```
    E = R - x;
    e(k) = E(1);
    de(k) = E(2);

    s(k) = Ce * E;

    X1 = abs(e(k)) + abs(de(k));

M = 2;
if M == 1                  % EXP reaching law
    ds(k) = - eq * ts * sign(s(k)) - q * ts * s(k);
    u(k) = inv(Ce * B) * (Ce * R1 - Ce * A * x - s(k) - ds(k));
elseif M == 2              % Variable rate reachine law
    ds(k) = - eq * ts * X1 * sign(s(k));
    u(k) = inv(Ce * B) * (Ce * R1 - Ce * A * x - s(k) - ds(k));
elseif M == 3              % Coposite reaching law
    k0 = 0.60;
    if X1>k0               % EXP reachine law
        ds(k) = - eq * ts * sign(s(k)) - q * ts * s(k);
        u(k) = inv(Ce * B) * (Ce * R1 - Ce * A * x - s(k) - ds(k));
    elseif X1< = k0        % Variable rate reachine law
    ds(k) = - eq * ts * X1 * sign(s(k));
    u(k) = inv(Ce * B) * (Ce * R1 - Ce * A * x - s(k) - ds(k));
    end
end

if u(k)> = 10
    u(k) = 10;
end
if u(k)< = - 10
    u(k) = - 10;
end

x = A * x + B * u(k);
y(k) = x(1);

% Update Parameters
r_2 = r_1;
r_1 = r(k);
end
figure(1)
plot(time,r,'r',time,y,'b');
xlabel('Time(second)');ylabel('Position tracking');
figure(2)
plot(time,s,'r');
xlabel('Time(second)');ylabel('Switch function s');
figure(3)
plot(e,de,'r',e, - c * e,'b');
xlabel('e');ylabel('de');
```

```
figure(4)
plot(time,u,'r');
xlabel('Time(second)');ylabel('u');
```

3.8　自适应离散滑模控制

利用离散趋近律设计离散系统的滑模变结构控制律，具有诸多优越性，但受到离散趋近律的参数和离散时间系统的采样周期的影响，系统会出现很大的抖振。参考文献[5]给出了对离散指数趋近律的抖振分析及控制律设计。

3.8.1　离散指数趋近律控制的抖振分析

基于指数的离散趋近律为

$$s(k+1)=(1-qT)s(k)-\varepsilon T\mathrm{sgn}(s(k)) \tag{3.66}$$

$$\begin{aligned}s(k+1)&=(1-qT)s(k)-\varepsilon T\frac{s(k)}{|s(k)|}\\&=\left(1-qT-\frac{\varepsilon T}{|s(k)|}\right)s(k)=ps(k)\end{aligned} \tag{3.67}$$

其中采样时间 T 很小，$T\ll 1.0$。

由式(3.67)可知

$$|p|=\frac{|s(k+1)|}{|s(k)|},\qquad p=1-qT-\frac{\varepsilon T}{|s(k)|} \tag{3.68}$$

显然 $p<1$。

针对式(3.67)，分以下三种情况进行讨论：

(1) 当 $|s(k)|>\dfrac{\varepsilon T}{2-qT}$ 时，有

$$p>1-Tq-\frac{\varepsilon T(2-qT)}{\varepsilon T}$$

$$p>-1$$

则 $|p|<1$，$|s(k+1)|<|s(k)|$，$|s(k)|$ 是递减的。

(2) 当 $|s(k)|<\dfrac{\varepsilon T}{2-qT}$ 时，有

$$p<1-Tq-\frac{\varepsilon T(2-qT)}{\varepsilon T}$$

$$p<-1$$

则 $|p|>1$，$|s(k+1)|>|s(k)|$，$|s(k)|$ 是递增的。

(3) 当 $|s(k)|=\dfrac{\varepsilon T}{2-qT}$ 时，有

$$p=1-Tq-\frac{\varepsilon T(2-qT)}{\varepsilon T}=-1$$

即$|p|=1$，$|s(k+1)|=|s(k)|$，$s(k)$进入振荡状态。

由上述分析可知，$s(k)$值递减的充分条件为

$$|s(k)| > \frac{\varepsilon T}{2-qT} \tag{3.69}$$

在滑模运动过程中，$|s(k)|$的值无限接近$\frac{\varepsilon T}{2-qT}$，一旦满足$|s(k)|=\frac{\varepsilon T}{2-qT}$，系统就进入振荡状态。对于任意初始值$s(0)\neq 0$，当$k\to\infty$，$|s(k)|\to\frac{\varepsilon T}{2-qT}$，且当$|s(k)|=\frac{\varepsilon T}{2-qT}$时，有$s(k+1)=-s(k)$。

因此，当$k\to\infty$时，滑模运动的稳态振荡幅度为

$$h=\frac{\varepsilon T}{2-qT} \tag{3.70}$$

可见，$s(k)$的收敛程度受ε、q和T的影响，尤其受ε、T的影响。只有当ε、T足够小时，$|s(k)|$才能变得很小。

3.8.2 自适应滑模控制器的设计

在离散趋近律式(3.66)中，参数ε的作用非常大，ε值减小，可降低系统的抖振。但ε值太小，影响系统到达切换面的趋近速度，同时由于技术、设备等因素，采样周期T也不可能取得很小。因此，理想的ε值应该是时变的，即系统运动开始时ε值应大一些，随着时间的增加，ε值应逐步减小。

由式(3.69)可知，只有使$|s(k)|>\frac{\varepsilon T}{2-qT}$时，$s(k)$值才会递减，这就要求$Tq+\frac{\varepsilon T}{|s(k)|}<2$，即$\varepsilon$的值应满足

$$\varepsilon < \frac{1}{T}(2-Tq)\,|s(k)| \tag{3.71}$$

取

$$\varepsilon = |s(k)|/2 \tag{3.72}$$

显然，如果采样时间T满足

$$T < \frac{4}{1+2q} \tag{3.73}$$

则式(3.71)得到满足。

由式(3.66)及式(3.72)得改进的离散趋近律为

$$s(k+1)-s(k) = -qTs(k)-\frac{|s(k)|}{2}T\mathrm{sgn}(s(k)) \tag{3.74}$$

参考式(3.14)，针对离散系统$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}u(k)$，假设$\boldsymbol{CB}\neq 0$。

离散趋近律式(3.74)所对应的控制律为

$$u(k) = -(\boldsymbol{CB})^{-1}\left[\boldsymbol{CA}\boldsymbol{x}(k)-(1-qT)s(k)+\frac{|s(k)|}{2}T\mathrm{sgn}(s(k))\right] \tag{3.75}$$

其中$s(k)=\boldsymbol{C}\boldsymbol{x}(k)$。

稳定性分析：

由离散趋近律式(3.74)得

$$[s(k+1)-s(k)]\mathrm{sgn}(s(k)) = \left[-qTs(k)-\frac{|s(k)|}{2}T\mathrm{sgn}(s(k))\right]\mathrm{sgn}(s(k))$$
$$=-(q+0.5)T|s(k)|<0 \tag{3.76}$$

$$[s(k+1)+s(k)]\mathrm{sgn}(s(k)) = \left[(2-qT)s(k)-\frac{|s(k)|}{2}T\mathrm{sgn}(s(k))\right]\mathrm{sgn}(s(k))$$
$$=(2-0.5T-qT)|s(k)|>0 \tag{3.77}$$

可见,滑模趋近律式(3.74)满足离散滑动模态的存在性和到达性条件,所设计的控制系统是稳定的。

3.8.3　仿真实例

对象传递函数为

$$G(s)=\frac{133}{s(s+25)}$$

采样周期为 0.001s,将对象离散化后的状态方程为

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}u(k)$$

其中 $\boldsymbol{A}=\begin{bmatrix}1 & 0.0010\\0 & 0.9753\end{bmatrix}$,$\boldsymbol{B}=\begin{bmatrix}0.0001\\0.1314\end{bmatrix}$。

指令取阶跃信号 $r(k)=1.0$,控制器参数为 $c=10,\varepsilon=15,q=30$,则滑模运动的稳态振荡幅度为 $h=\frac{\varepsilon T}{2-qT}=0.0076$,$\boldsymbol{C}=[c,1]$。

被控对象初值取[−0.5,−0.5],分别采用指数趋近律、自适应趋近律进行仿真。

$M=1$ 时,控制律为指数趋近律,采用式(3.44),仿真结果如图 3-39～图 3-42 所示。由仿真结果可知,$|s(k)|$的值在$[-h,+h]$范围内,和式(3.70)相符。$M=2$ 时,采用自适应趋近律,控制器采用式(3.75),仿真结果如图 3-43～图 3-46 所示。系统状态并不出现振荡,而是渐近地趋向平衡点零,控制输入信号平滑无抖振。

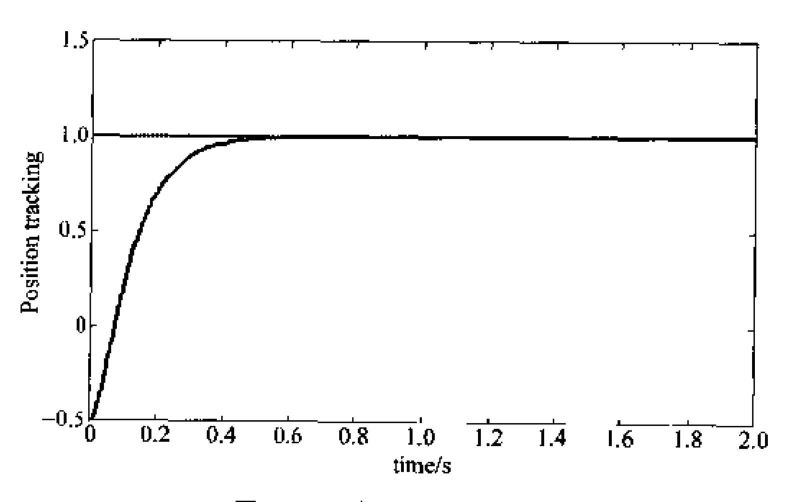


图 3-39　阶跃响应($M=1$)

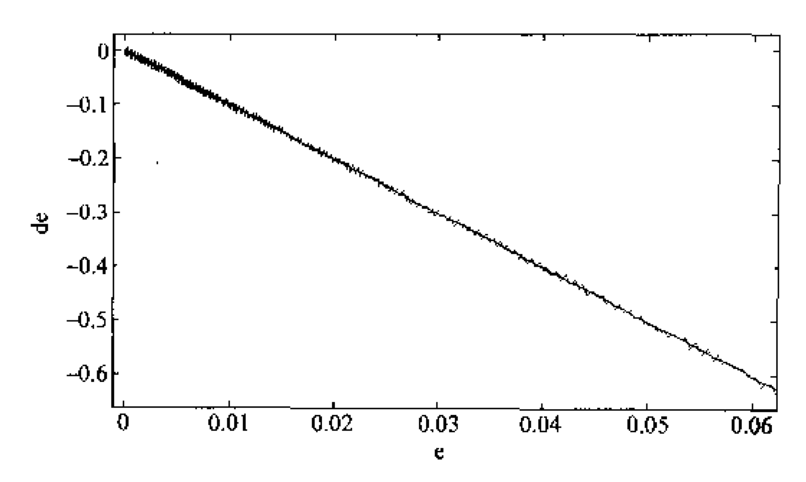


图 3-40 相轨迹变化的局部放大（$M=1$）

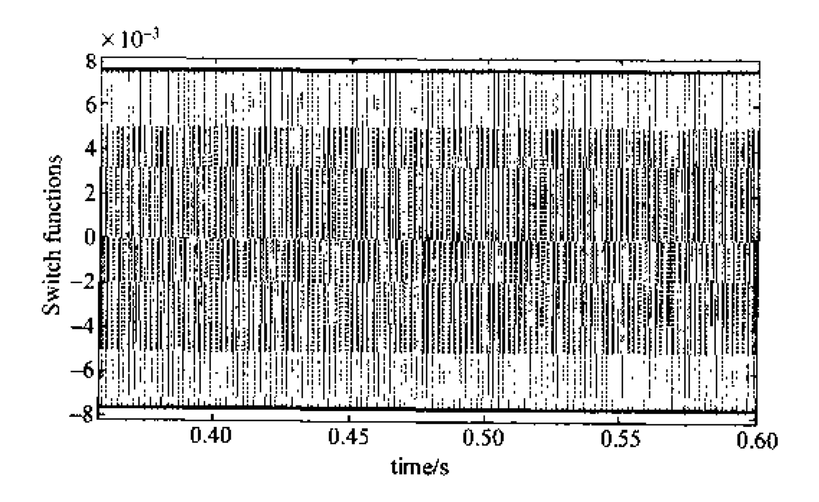


图 3-41 切换函数变化的局部放大（$M=1$）

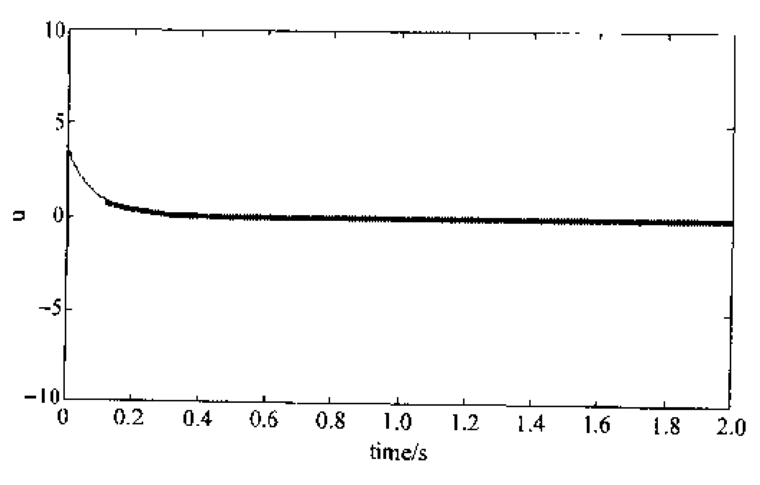


图 3-42 控制器输出（$M=1$）

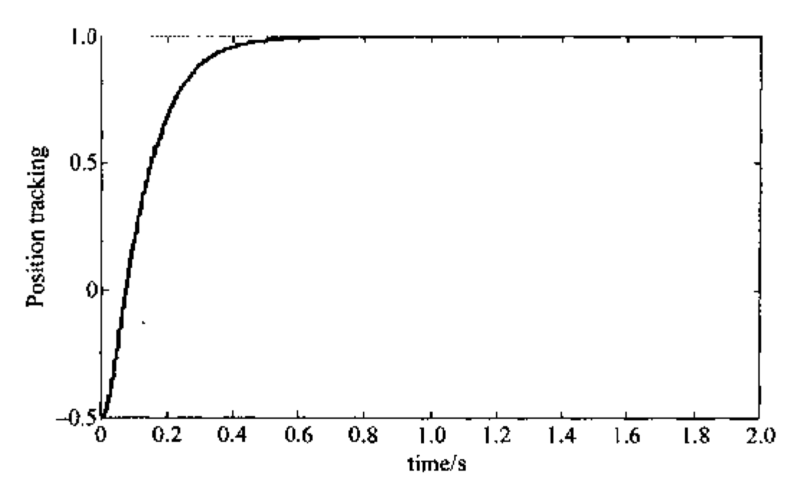


图 3-43　阶跃响应（$M=2$）

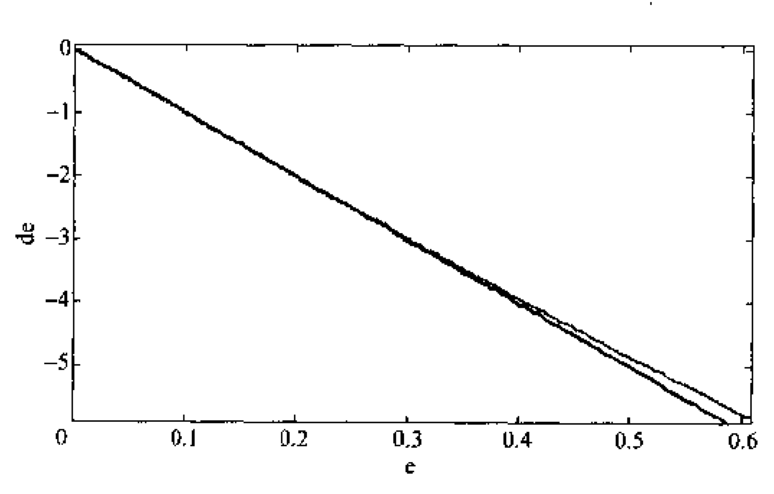


图 3-44　相轨迹变化的局部放大（$M=2$）

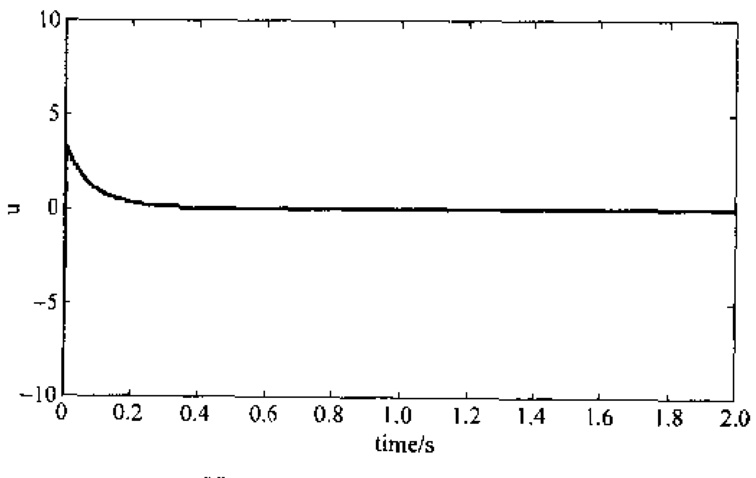


图 3-45　控制器输出（$M=2$）

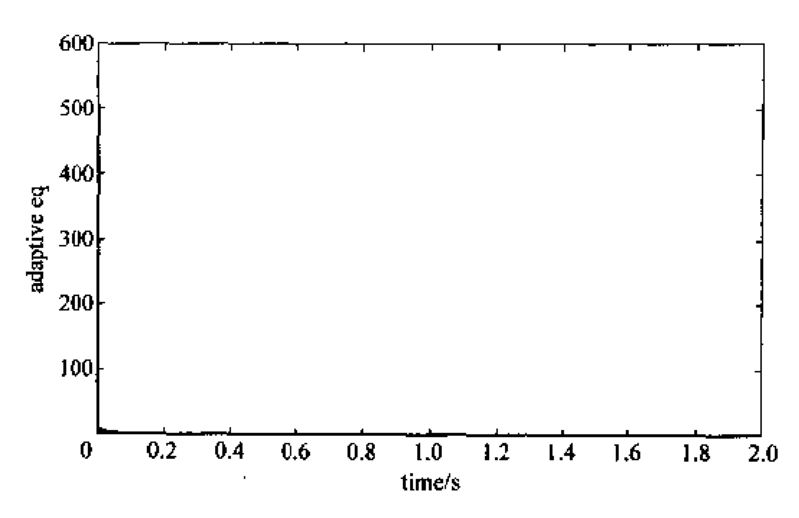


图 3-46 ε 值的自适应的变化（$M=2$）

仿真程序：chap3_8.m

```
clear all;
close all;

a = 25;b = 133;

ts = 0.001;
A1 = [0,1;0, - a];
B1 = [0;b];
C1 = [1,0];
D1 = 0;
[A,B,C,D] = c2dm(A1,B1,C1,D1,ts,'z');

x = [ - 0.5; - 0.5];
r_1 = 0;r_2 = 0;

c = 10;
q = 30;
Ce = [c,1];
for k = 1:1:2000

   time(k) = k * ts;

  r(k) = 1.0;

  % Using Waitui method
  dr(k) = (r(k) - r_1)/ts;
  dr_1 = (r_1 - r_2)/ts;
  r1(k) = 2 * r(k) - r_1;
  dr1(k) = 2 * dr(k) - dr_1;

  R = [r(k);dr(k)];
  R1 = [r1(k);dr1(k)];
```

```
    E = R - x;
    e(k) = E(1);
    de(k) = E(2);

    s(k) = Ce * E;

 M = 2;
 if M == 1                   % EXP reaching law
    eq(k) = 15;
    ds(k) = - eq(k) * ts * sign(s(k)) - q * ts * s(k);
    u(k) = inv(Ce * B) * (Ce * R1 - Ce * A * x - s(k) - ds(k));
 elseif M == 2               % Adaptive trending law
    eq(k) = abs(s(k))/2;
    ds(k) = - eq(k) * ts * sign(s(k)) - q * ts * s(k);
    u(k) = inv(Ce * B) * (Ce * R1 - Ce * A * x - s(k) - ds(k));
 end
     h(k) = eq(k) * ts/(2 - q * ts);

 if u(k)> = 10
    u(k) = 10;
 end
 if u(k)< = - 10
    u(k) = - 10;
 end

x = A * x + B * u(k);
y(k) = x(1);

 % Update Parameters
r_2 = r_1;
r_1 = r(k);
end
figure(1)
plot(time,r,'r',time,y,'b');
xlabel(' Time(second)');ylabel('position tracking)');

figure(2)
plot(time,s,'r',time,h,'b',time, - h,'b');
xlabel(' Time(second)');ylabel('Switch function s');

figure(3)
plot(e,de,'r',e, - c * e,'b');
xlabel('e');ylabel('de');

figure(4)
plot(time,u,'r');
xlabel(' Time(second)');ylabel('u');

figure(5);
plot(time,eq,'r');
xlabel('time(s)');ylabel('adaptive eq');
```

参 考 文 献

1. Gao W B, Wang Y F, Homaifa A. Discrete-time variable structure control systems. IEEE Transactions on Industrial Electronics, 1995, 42(2): 117～122
2. Furuta K. Sliding mode control of a discrete system. Systems & Control Letters, 1990, 14(2): 145～152
3. 宋立忠,温洪,姚琼荟. 离散变结构控制系统的变速趋近律. 海军工程学院学报, 1999, 3: 16～21
4. Sarpturk S Z, Istefanopulos Y, Kaynak O. On the stability of discrete-time sliding mode control system. IEEE Transactions on Automatic Control, 1987, 32(10): 930～932
5. 翟长连,吴智铭. 不确定离散时间系统的变结构控制设计. 自动化学报, 2000, 26(2): 184～191
6. 邹云飞,刘金琨,王宗学. 一种离散变结构控制方法在伺服系统中的应用. 机床与液压, 2002, 2: 31～33

第4章 模糊滑模控制

4.1 常规模糊滑模控制

4.1.1 基本原理

20世纪90年代后，将滑模变结构控制与自适应控制、干扰补偿、神经元网络及模糊控制等相结合构成新型控制方法的研究已成为新的热点。模糊控制(fuzzy logic control)以20世纪60年代Zadeh的模糊数学为基本理论基础，从70年代起进入实际工程应用阶段。在过去的20多年中，模糊控制作为一种有别于传统控制理论的控制方法，充分发挥其不需要对象数学模型、能充分运用控制专家的信息及具有鲁棒性的优点，在具有相关特点的控制领域表现出其优势。在一些复杂系统，特别是系统存在不精确和不确定信息的情况下，模糊控制的效果往往优于常规控制。另一方面，一般的实用模糊控制器仍有其需要面对的问题，即模糊控制器参数必须经过反复试凑才能确定，缺少稳定性分析等系统化的分析和综合方法。

模糊滑模控制(fuzzy sliding mode control)是将模糊控制和传统的滑模控制相结合，并将两者的优点紧密结合起来。模糊滑模控制保持了常规模糊控制器的优点，即可以不依赖系统的模型。但是模糊滑模控制相对于常规模糊控制的变化具有两个方面的重要意义，一是控制目标从跟踪误差转为滑模函数，只要施加控制使滑模函数 s 为零，跟踪误差将渐近到达零点；二是对于 $n>2$ 的高阶系统，在常规模糊控制中输入应为 $[e,\dot{e},\cdots,e^{(n-1)}]$，而模糊滑模控制的输入 $(s,\dot{s})$ 始终是二维的。总之，在 $n>2$ 的特定情况下，模糊滑模控制具有简化模糊控制系统结构复杂性的作用。对于滑模控制而言，模糊滑模控制的意义在于它柔化控制信号，减轻或避免了一般滑模控制的抖振现象。

采用控制的变化量 Δu 作为模糊滑模控制器的输出，可使模糊滑模控制成为无模型控制，依赖于被控对象的程度小。

4.1.2 模糊滑模控制器的设计

为了克服参数不确定性及外干扰的影响，满足滑模运动存在的条件，并且削弱滑模控制的抖振程度，根据前人的经验，可利用模糊控制规则调整控制变量 u 的大小，确保条件 $s\dot{s}<0$ 成立[1]。

设采样时间为 T，则

$$e(k)=r(k)-y(k),\mathrm{d}e(k)=\frac{e(k)-e(k-1)}{T}$$

切换函数为

$$s(k)=ce(k)+\mathrm{d}e(k)=\boldsymbol{CX}(k) \tag{4.1}$$

$$ds(k)=s(k)-s(k-1) \tag{4.2}$$

其中 $\boldsymbol{C}=[c,1]$。

采用二维模糊控制器，通过模糊控制规律直接设计滑模控制量 u。设模糊控制器的输入是 s 和 $\dot{s}$，它们分别是 $s(k)$ 和 $ds(k)$ 的模糊化变量，模糊控制器的输出 ΔU 是控制的变化量 Δu 的模糊化变量。

（1）定义模糊集

PB＝正大 PM＝正中 PS＝正小

NS＝负小 NM＝负中 NB＝负大

（2）根据模糊控制原理，定义 s 和 $\dot{s}$ 为模糊控制器的输入，输出为 ΔU

$s=\{$NB,NM,NS,ZO,PS,PM,PB$\}$

$\dot{s}=\{$NB,NM,NS,ZO,PS,PM,PB$\}$

$\Delta U=\{$NB,NM,NS,ZO,PS,PM,PB$\}$

其论域为

$$s=\{-3,-2,-1,0,+1,+2,+3\}$$

$$\dot{s}=\{-3,-2,-1,0,+1,+2,+3\}$$

$$\Delta U=\{-3,-2,-1,0,+1,+2,+3\}$$

上述模糊化变量均选择正态分布隶属函数。

（3）确定模糊滑模控制器的模糊控制规则

在力图满足不等式 $s\dot{s}<0$ 的条件下设计 u，所获得的控制规则如表 4-1 所示。使用的模糊规则是：If s is A and $\dot{s}$ is B, then ΔU is C。

表 4-1 模糊滑模控制规则表

s \ $\dot{s}$	NB	NM	NS	ZO	PS	PM	PB
PB	ZO	PS	PM	PB	PB	PB	PB
PM	NS	ZO	PS	PM	PB	PB	PB
PS	NM	NS	ZO	PS	PM	PB	PB
ZO	NB	NM	NS	ZO	PS	PM	PB
NS	NB	NB	NM	NS	ZO	PS	PM
NM	NB	NB	NB	NM	NS	ZO	PS
NB	NB	NB	NB	NB	NM	NS	ZO

从表 4-1 可看出，当 s 和 $\dot{s}$ 都为正大，这意味着 $s\dot{s}$ 是正大，那么就需要控制输入一个大的正的变化，以使 $s\dot{s}$ 快速减小（If s is PB and $\dot{s}$ is PB, then ΔU is PB）；当 $s\dot{s}<0$ 时，为期望的状态，控制变化量为零（If s is PB and $\dot{s}$ is NB, then ΔU is ZO）；当 s 和 $\dot{s}$ 都为负大，这意味着 $s\dot{s}$ 是正大，那么需要控制输入一个大的负的变化，以使 $s\dot{s}$ 快速减小（If s is NB and $\dot{s}$ is NB, then ΔU is NB）。由于表 4-1 中所有的控制规则是根据满足 $s\dot{s}<0$ 这个达到滑模的充要条件所设计的，所以设计的模糊滑模控制系统是稳定的。

（4）反模糊化：采用重心法将模糊输出精确化，公式如下：

$$uf(k)=\frac{\sum_{i=1}^{n}x_i\mu(i)}{\sum_{i=1}^{n}\mu(i)} \tag{4.3}$$

4.1.3 仿真实例

对象传递函数为

$$G(s)=\frac{133}{s(s+25)}$$

采样时间为 1ms,采用 Z 变换进行离散化,经过 Z 变换后的离散化对象为

$$\begin{aligned}\text{yout}(k)=&-\text{den}(2)\text{yout}(k-1)-\text{den}(3)\text{yout}(k-2)-\text{den}(4)\text{yout}(k-3)\\&+\text{num}(2)u(k-1)+\text{num}(3)u(k-2)+\text{num}(4)u(k-3)\end{aligned}$$

其中 yout 为离散化后被控对象的输出,k 为采样次数,u 为控制输入。

首先建立模糊系统 fsmc. fis,模糊系统的输入输出值范围均取[－3,3],运行 showrule(a)可显示模糊系统的规则库,运行 ruleview(a)可进入模糊系统的动态调试环境。程序运行结果可显示模糊系统输入输出的隶属函数图,见图 4-1 和图 4-2。

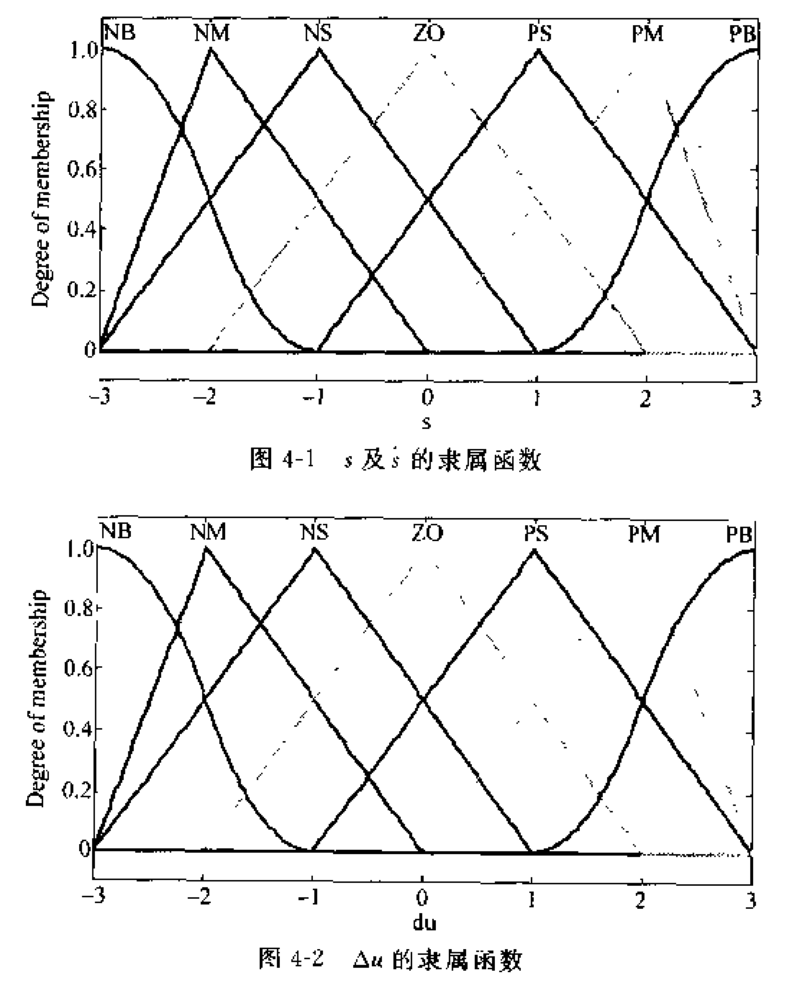


图 4-1　s 及 $\dot{s}$ 的隶属函数

图 4-2　Δu 的隶属函数

滑模参数取 $c=15$,控制输入信号限制在[－10,10]。取 $M=1$,跟踪阶跃信号,仿真结

果如图4-3～图4-5所示。取$M=2$，跟踪幅值为1.0、频率为1.0的方波信号，仿真结果如图4-6所示。取$M=3$，跟踪幅值为1.0、频率为2.0的正弦信号，仿真结果如图4-7所示。

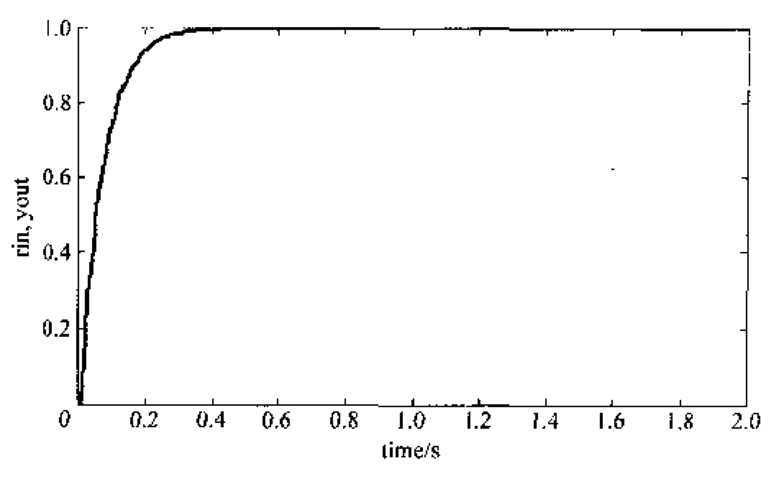


图4-3 阶跃响应($M=1$)

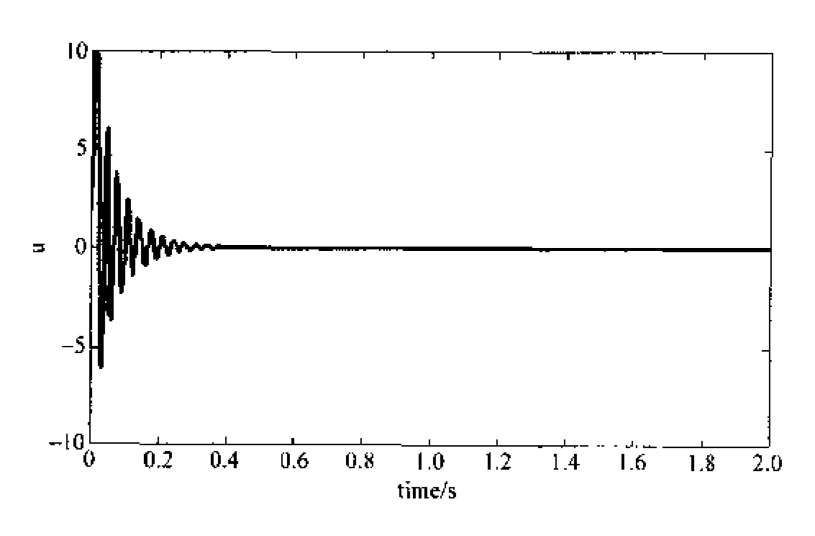


图4-4 控制曲线($M=1$)

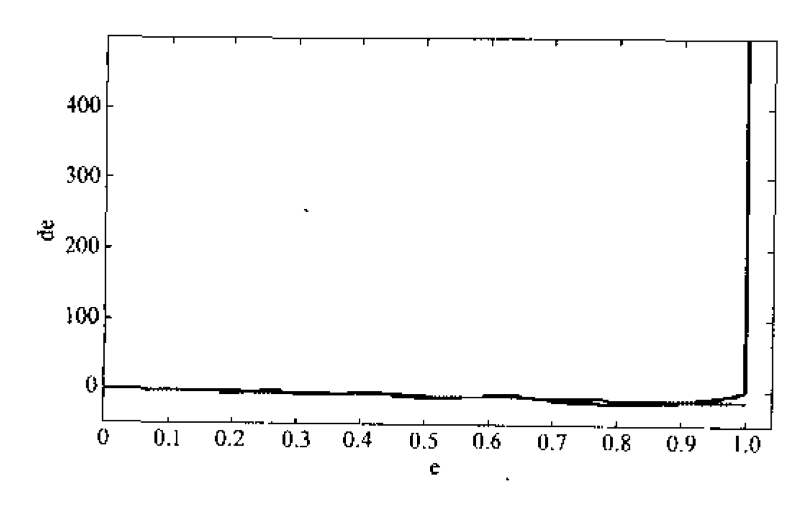


图4-5 运动相轨迹($M=1$)

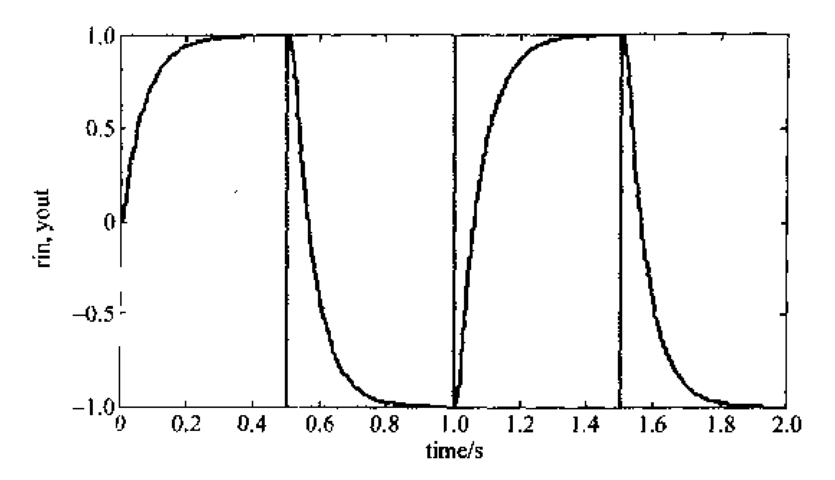


图 4-6　方波响应($M=2$)

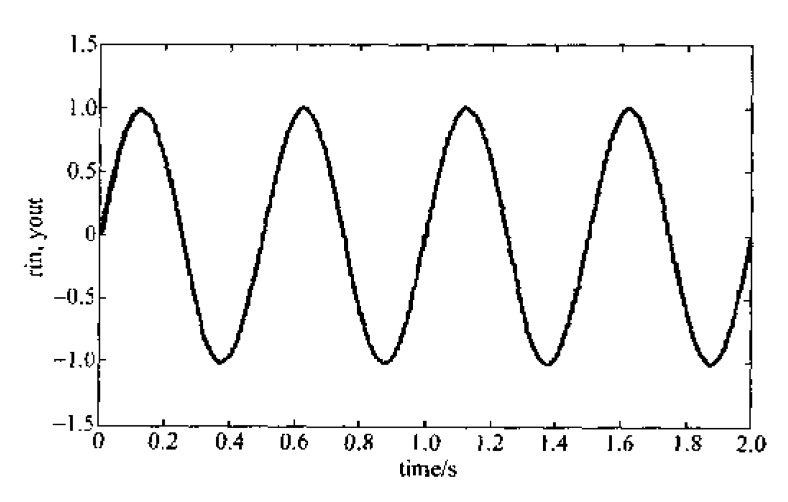


图 4-7　正弦响应($M=3$)

仿真程序：chap4_1.m

```
clear all;
close all;

%%%%%%%%%%%%%%%%%%%%%% Fuzzy System Design %%%%%%%%%%%%%%%%%%%%%%
a = newfis('fuzz_smc');

k1 = 1.0;
a = addvar(a,'input','s',[-3*k1,3*k1]); % Parameter s
a = addmf(a,'input',1,'NB','zmf',[-3*k1,-1*k1]);
a = addmf(a,'input',1,'NM','trimf',[-3*k1,-2*k1,0]);
a = addmf(a,'input',1,'NS','trimf',[-3*k1,-1*k1,1*k1]);
a = addmf(a,'input',1,'Z','trimf',[-2*k1,0,2*k1]);
a = addmf(a,'input',1,'PS','trimf',[-1*k1,1*k1,3*k1]);
a = addmf(a,'input',1,'PM','trimf',[0,2*k1,3*k1]);
a = addmf(a,'input',1,'PB','smf',[1*k1,3*k1]);

k2 = 1.0;
a = addvar(a,'input','ds',[-3*k2,3*k2]); % Parameter ds
```

```
a = addmf(a,'input',2,'NB','zmf',[ - 3 * k2, - 1 * k2]);
a = addmf(a,'input',2,'NM','trimf',[ - 3 * k2, - 2 * k2,0]);
a = addmf(a,'input',2,'NS','trimf',[ - 3 * k2, - 1 * k2,1 * k2]);
a = addmf(a,'input',2,'Z','trimf',[ - 2 * k2,0,2 * k2]);
a = addmf(a,'input',2,'PS','trimf',[ - 1 * k2,1 * k2,3 * k2]);
a = addmf(a,'input',2,'PM','trimf',[0,2 * k2,3 * k2]);
a = addmf(a,'input',2,'PB','smf',[1 * k2,3 * k2]);

k3 = 1.0;
a = addvar(a,'output','du',[ - 3 * k3,3 * k3]); % Parameter du
a = addmf(a,'output',1,'NB','zmf',[ - 3 * k3, - 1 * k3]);
a = addmf(a,'output',1,'NM','trimf',[ - 3 * k3, - 2 * k3,0]);
a = addmf(a,'output',1,'NS','trimf',[ - 3 * k3, - 1 * k3,1 * k3]);
a = addmf(a,'output',1,'Z','trimf',[ - 2 * k3,0,2 * k3]);
a = addmf(a,'output',1,'PS','trimf',[ - 1 * k3,1 * k3,3 * k3]);
a = addmf(a,'output',1,'PM','trimf',[0,2 * k3,3 * k3]);
a = addmf(a,'output',1,'PB','smf',[1 * k3,3 * k3]);

rulelist = [1 1 1 1 1; % Fuzzy rule base
          1 2 1 1 1;
          1 3 2 1 1;
          1 4 2 1 1;
          1 5 3 1 1;
          1 6 3 1 1;
          1 7 4 1 1;

          2 1 1 1 1;
          2 2 2 1 1;
          2 3 2 1 1;
          2 4 3 1 1;
          2 5 3 1 1;
          2 6 4 1 1;
          2 7 5 1 1;

          3 1 2 1 1;
          3 2 2 1 1;
          3 3 3 1 1;
          3 4 3 1 1;
          3 5 4 1 1;
          3 6 5 1 1;
          3 7 5 1 1;

          4 1 2 1 1;
          4 2 3 1 1;
          4 3 3 1 1;
          4 4 4 1 1;
          4 5 5 1 1;
          4 6 5 1 1;
          4 7 6 1 1;

          5 1 3 1 1;
```

```
        5 2 3 1 1;
        5 3 4 1 1;
        5 4 5 1 1;
        5 5 5 1 1;
        5 6 6 1 1;
        5 7 6 1 1;

        6 1 3 1 1;
        6 2 4 1 1;
        6 3 5 1 1;
        6 4 5 1 1;
        6 5 6 1 1;
        6 6 6 1 1;
        6 7 7 1 1;

        7 1 4 1 1;
        7 2 5 1 1;
        7 3 5 1 1;
        7 4 6 1 1;
        7 5 6 1 1;
        7 6 7 1 1;
        7 7 7 1 1];

a = addrule(a,rulelist);
%showrule(a)              % Show fuzzy rule base
%ruleview(a)
a1 = setfis(a,'DefuzzMethod','mom'); % Defuzzy
writefis(a1,'fsmc');               % Save to fuzzy file "fsmc.fis"
a2 = readfis('fsmc');

%%%%%%%%%%%%%%%%%%%%%% Fuzzy SMC Control %%%%%%%%%%%%%%%%%%%%%
ts = 0.001;
sys = tf([133],[1,25,0]);
dsys = c2d(sys,ts,'z');
[num,den] = tfdata(dsys,'v');

e_1 = 0;de_1 = 0;s_1 = 0;
u_1 = 0.0;u_2 = 0.0;
y_1 = 0.0;y_2 = 0.0;

for k = 1:1:2000
time(k) = k * ts;

M = 1;
if M == 1
  rin(k) = 1;  % Step Signal
end
if M == 2
  rin(k) = sign(sin(1 * 2 * pi * k * ts));  % Square Signal
end
if M == 3
```

```
  rin(k) = sin(2 * 2 * pi * k * ts);  % Sine Signal
end

% Linear model
yout(k) = - den(2) * y_1 - den(3) * y_2 + num(2) * u_1 + num(3) * u_2;

e(k) = rin(k) - yout(k);
de(k) = (e(k) - e_1)/ts;

c = 15;
s(k) = c * e(k) + 1 * de(k);
s1(k) = s(k) - s_1;

du(k) = evalfis([s(k) s1(k)],a2);  % Fuzzy Slide Mode Controler
u(k) = u_1 + du(k);

if u(k)> = 10
  u(k) = 10;
end
if u(k)< = - 10
  u(k) = - 10;
end

e_1 = e(k);
de_1 = de(k);
s_1 = s(k);

u_2 = u_1;u_1 = u(k);
y_2 = y_1;y_1 = yout(k);
end
figure(1);
plot(time,rin,'r',time,yout,'b');
xlabel('time(s)');ylabel('rin,yout');
figure(2);
plot(e,de,'b',e, - c * e,'r');
xlabel('e');ylabel('de');
figure(3);
plot(time,s,'r');
xlabel('time(s)');ylabel('s');
figure(4);
plot(time,u,'r');
xlabel('time(s)');ylabel('u');

figure(5);
plotmf(a2,'input',1);
figure(6);
plotmf(a2,'input',2);
figure(7);
plotmf(a2,'output',1);
```

4.2　基于模糊自适应调节的滑模控制

4.2.1　模糊自适应调节原理

设离散系统状态方程如下：

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}u(k) \qquad \boldsymbol{x}\in\mathbf{R}^n, u\in\mathbf{R} \tag{4.4}$$

设位置指令为 $r(k)$，位置指令的导数为 $\mathrm{d}r(k)$，$\boldsymbol{R}=[r(k),\mathrm{d}r(k)]$，$\boldsymbol{R}_1=[r(k+1),\mathrm{d}r(k+1)]$。$r(k+1)$及 $\mathrm{d}r(k+1)$采用线性外推的方法进行预测：

$$r(k+1)=2r(k)-r(k-1), \mathrm{d}r(k+1)=2\mathrm{d}r(k)-\mathrm{d}r(k-1)$$

切换函数为

$$s(k)=\boldsymbol{C}_e\boldsymbol{E}=\boldsymbol{C}_e(\boldsymbol{R}(k)-\boldsymbol{x}(k))$$

其中 $\boldsymbol{C}_e=[c,1]$。

由式(3.45)可知，基于指数趋近律的离散控制律：

$$u(k)=(\boldsymbol{C}_e\boldsymbol{B})^{-1}(\boldsymbol{C}_e\boldsymbol{R}(k+1)-\boldsymbol{C}_e\boldsymbol{A}\boldsymbol{x}(k)-s(k)-\mathrm{d}s(k)) \tag{4.5}$$

其中 $\mathrm{d}s(k)=-\varepsilon T\mathrm{sgn}(s(k))-qTs(k)$。

在采样时间固定的条件下，ε 的值决定了控制器抖振的幅度。取 ε 为模糊控制系统的输出 $fs(k)$的绝对值：

$$\varepsilon=|fs(k)| \tag{4.6}$$

设计二输入单输出模糊控制器，取切换函数 $s(k)$的误差及其变化率 $ds(k)$作为输入，变化范围为[-3,3]，$fs(k)$作为输出，变化范围为[-3,3]。通过采用模糊控制规则和模糊推理使 $s(k)\to 0$。

根据经验，采用以下 9 条模糊规则：

(1) If (s is NB) and (ds is NB) then (fs is NB)

(2) If (s is NB) and (ds is ZO) then (fs is NB)

(3) If (s is NB) and (ds is PB) then (fs is ZO)

(4) If (s is ZO) and (ds is NB) then (fs is NB)

(5) If (s is ZO) and (ds is ZO) then (fs is ZO)

(6) If (s is ZO) and (ds is PB) then (fs is ZO)

(7) If (s is PB) and (ds is NB) then (fs is ZO)

(8) If (s is PB) and (ds is ZO) then (fs is ZO)

(9) If (s is PB) and (ds is PB) then (fs is PB)

模糊系统的输入和输出隶属函数如图 4-8 和图 4-9 所示。

4.2.2　仿真实例

对象传递函数为

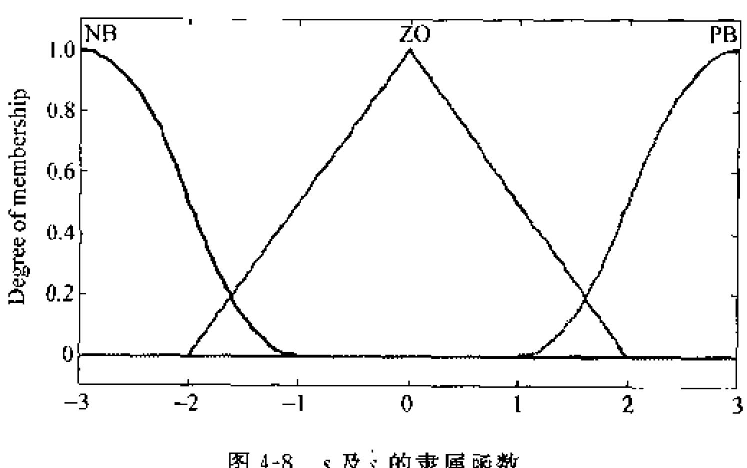


图 4-8　s 及 $\dot{s}$ 的隶属函数

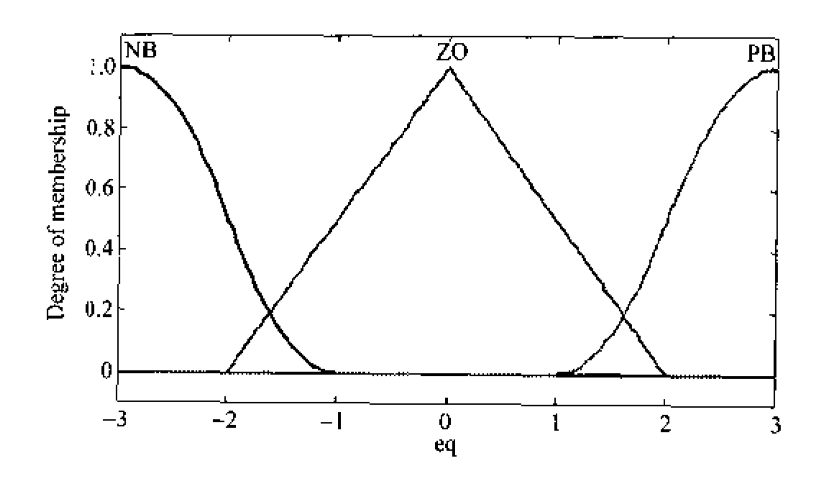


图 4-9　fs 的隶属函数

$$G(s)=\frac{133}{s(s+25)}$$

控制系统的采样周期为 0.001s，将系统离散化后的状态方程为

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}u(k)$$

其中 $\boldsymbol{A}=\begin{bmatrix}1 & 0.0010\\0 & 0.9753\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0.0001\\0.1314\end{bmatrix}$。

首先建立模糊系统 fuzz.fis，模糊系统的输入输出值范围均取[-3,3]，运行 showrule(a)可显示模糊系统的规则库，运行 ruleview(a)可进入模糊系统的动态调试环境。程序运行结果可显示模糊系统输入输出的隶属函数图，见图 4-8 和图 4-9。

指令取阶跃信号 $r(k)=1.0$，控制器参数为 $c=10$，$q=30$。

被控对象初值取[0,0]，控制输入信号限制在[-10,10]。分别采用指数趋近律、模糊趋近律进行仿真。控制器中，取 $c=25$，$q=50$，$M=1$ 为采用 $\varepsilon=60$ 的指数趋近律控制，$M=2$ 为采用模糊自适应调整 ε 的指数趋近律控制。当 $M=1$ 时，仿真结果如图 4-10 和图 4-11 所示，当 $M=2$ 时，仿真结果如图 4-12～图 4-14 所示。可见，采用自适应趋近律滑模控制，可以大大降低控制的抖振。

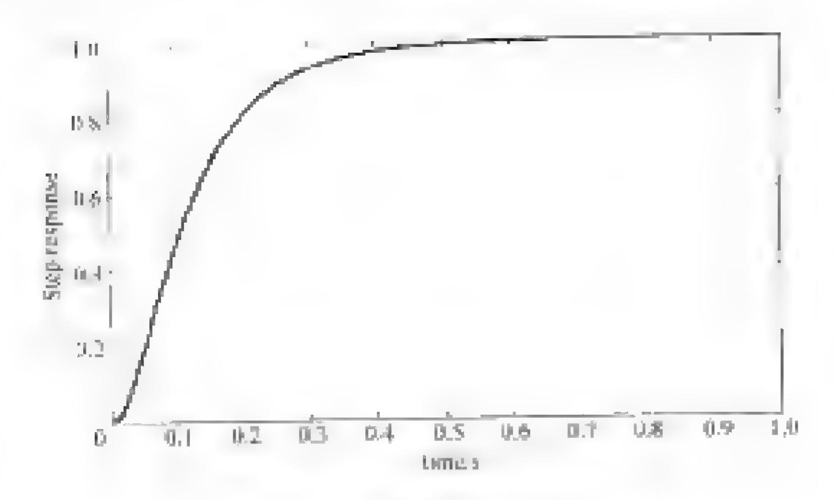


图 4-10 阶跃响应 ($M=1$)

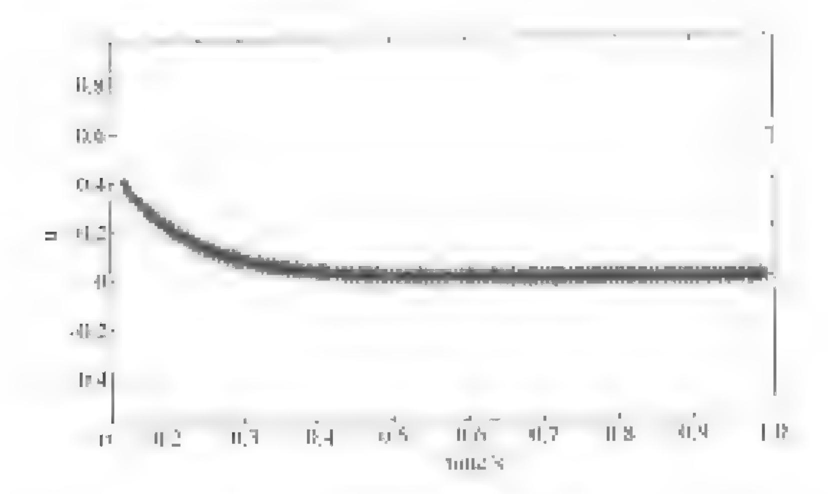


图 4-11 控制器输出放大图 ($M=1$)

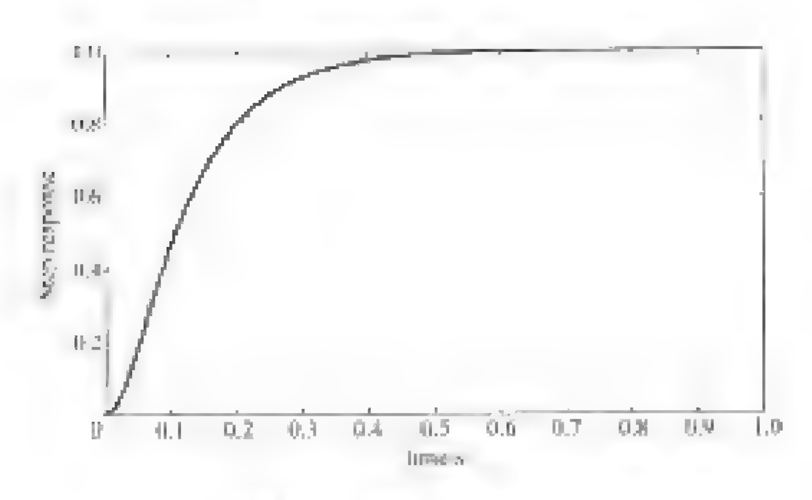


图 4-12 阶跃响应 ($M=2$)

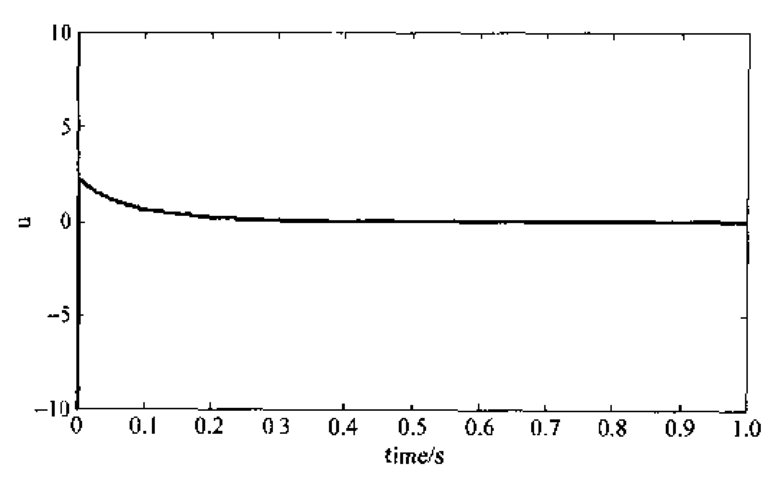


图 4-13 控制器输出（$M=2$）

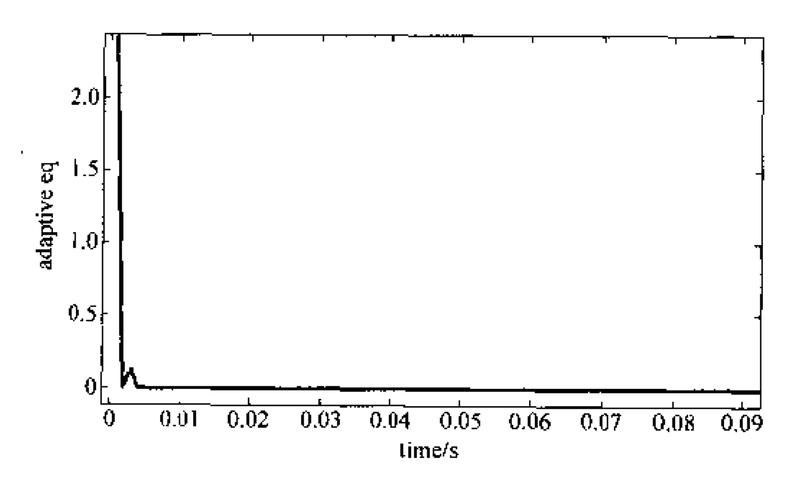


图 4-14 ε 的变化过程放大图（$M=2$）

仿真程序：chap4_2.m

```
clear all;
close all;
%%%%%%%% Fuzzy System Design %%%%%%%%%%%
a = newfis('fuzz');

k1 = 1.0;
a = addvar(a,'input','s',[-3*k1,3*k1]);
a = addmf(a,'input',1,'NB','zmf',[-3*k1,-1*k1]);
a = addmf(a,'input',1,'Z','trimf',[-2*k1,0,2*k1]);
a = addmf(a,'input',1,'PB','smf',[1*k1,3*k1]);

k2 = 1.0;
a = addvar(a,'input','ds',[-3*k2,3*k2]);
a = addmf(a,'input',2,'NB','zmf',[-3*k2,-1*k2]);
a = addmf(a,'input',2,'Z','trimf',[-2*k2,0,2*k2]);
a = addmf(a,'input',2,'PB','smf',[1*k2,3*k2]);

k3 = 1.0;
a = addvar(a,'output','eq',[-3*k3,3*k3]);
```

```
a = addmf(a,'output',1,'NB','zmf',[-3*k3,-1*k3]);
a = addmf(a,'output',1,'Z','trimf',[-2*k3,0,2*k3]);
a = addmf(a,'output',1,'PB','smf',[1*k3,3*k3]);

rulelist = [1 1 1 1 1; % Edit rule base
         1 2 1 1 1;
         1 3 2 1 1;

         2 1 1 1 1;
         2 2 2 1 1;
         2 3 2 1 1;

         3 1 2 1 1;
         3 2 2 1 1;
         3 3 3 1 1];

a = addrule(a,rulelist);
 %showrule(a) % Show fuzzy rule base
 %ruleview(a)
a1 = setfis(a,'DefuzzMethod','mom'); % Defuzzy
writefis(a1,'fuzz');
a2 = readfis('fuzz');

 %%%%%%%% Fuzzy SMC Control %%%%%%%%%%
ts = 0.001;
A1 = [0,1;0,-25];
B1 = [0;133];
C1 = [1,0];
D1 = 0;
[A,B,C,D] = c2dm(A1,B1,C1,D1,ts,'z');

x = [0;0];
r_1 = 0;r_2 = 0;

c = 10;
q = 30;
Ce = [c,1];
es_1 = 0;

for k = 1:1:1000

   time(k) = k*ts;
   r(k) = 1.0;

   %Using Waitui method
   dr(k) = (r(k) - r_1)/ts;
   dr_1 = (r_1 - r_2)/ts;
   r1(k) = 2*r(k) - r_1;
   dr1(k) = 2*dr(k) - dr_1;

R = [r(k);dr(k)];
```

```
R1 = [r1(k);dr1(k)];

E = R - x;
e(k) = E(1);
de(k) = E(2);

s(k) = Ce * E;

es(k) = s(k);
des(k) = es(k) - es_1;

fs(k) = evalfis([es(k) des(k)],a2); % Using fuzzy inference

M = 2;
if M == 1 % EXP trending law with fixed eq
  eq(k) = 5.0;
elseif M == 2 % EXP trending law with adaptive eq
  eq(k) = abs(fs(k));
end

ds(k) = - eq(k) * ts * sign(s(k)) - q * ts * s(k);
u(k) = inv(Ce * B) * (Ce * R1 - Ce * A * x - s(k) - ds(k));

if u(k)>= 10
  u(k) = 10;
end
if u(k)<= - 10
  u(k) = - 10;
end

x = A * x + B * u(k);
y(k) = x(1);

% Update Parameters
r_2 = r_1;
r_1 = r(k);
es_1 = es(k);
end
figure(1)
plot(time,r,'r',time,y,'b');
xlabel('Time(second)');ylabel('Step response');

figure(2)
plot(e,de,'r',e, - c * e,'b');
xlabel('e');ylabel('de');

figure(3)
plot(time,u,'r');
xlabel('Time(second)');ylabel('u');

figure(4)
```

```
plot(time,eq,'r');
xlabel('time(s)');ylabel('adaptive eq');

figure(5);
plotmf(a2,'input',1);
figure(6);
plotmf(a2,'input',2);
figure(7);
plotmf(a2,'output',1);
```

4.3　基于模糊切换增益调节的滑模控制

采用模糊规则,可根据滑模到达条件对切换增益进行有效的估计,并利用切换增益消除干扰项,从而消除抖振[2]。

4.3.1　系统描述

考虑不确定伺服系统

$$\dot{\boldsymbol{x}}=(\boldsymbol{A}+\Delta\boldsymbol{A})\boldsymbol{x}+(\boldsymbol{B}+\Delta\boldsymbol{B})u(t)+(\boldsymbol{d}+\Delta\boldsymbol{d})f \tag{4.7}$$

其中 $\boldsymbol{x}\in\mathbf{R}^n,u\in\mathbf{R},f\in\mathbf{R},\boldsymbol{A},\boldsymbol{B}$ 为已知,$\Delta\boldsymbol{A}$ 和 $\Delta\boldsymbol{B}$ 为参数变化,$\boldsymbol{d}$ 为外加干扰。

假设系统满足匹配条件:

$$\Delta\boldsymbol{A}=\boldsymbol{B}\tilde{\boldsymbol{A}},\Delta\boldsymbol{B}=\boldsymbol{B}\tilde{\boldsymbol{B}},d+\Delta d=\boldsymbol{B}\tilde{d} \tag{4.8}$$

则

$$\dot{\boldsymbol{x}}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}[u(t)+E(t)] \tag{4.9}$$

其中 $E(t)$包括不确定和外加干扰。

$$E(t)=\tilde{\boldsymbol{A}}\boldsymbol{x}+\tilde{\boldsymbol{B}}u(t)+\tilde{\boldsymbol{d}}f \tag{4.10}$$

取

$$\boldsymbol{x}=\begin{bmatrix}\theta\\\dot{\theta}\end{bmatrix},\boldsymbol{A}=\begin{bmatrix}0&1\\0&-a\end{bmatrix},\boldsymbol{B}=\begin{bmatrix}0\\b\end{bmatrix} \tag{4.11}$$

其中 θ 和 $\dot{\theta}$ 分别为角度和角速度。

状态方程式(4.9)可描述为

$$\ddot{\theta}=f(\theta)+b[u(t)+E(t)] \tag{4.12}$$

其中 $f(\theta)=-a\dot{\theta}$。

假设

$$K(t)=\max(|E(t)|)+\eta \tag{4.13}$$

其中 $\eta>0$。

4.3.2　滑模控制器设计

定义全局滑模面为

$$s=\dot{e}+ce-F(t) \tag{4.14}$$

其中 $c>0$，e 为跟踪误差。

跟踪误差为

$$e=r-\theta \tag{4.15}$$

其中 r 为位置指令。

为了实现全局滑模，函数 $F(t)$ 需要满足以下三个条件：

(1) $F(0)=\dot{e}_0+ce_0$；

(2) $F(t)\to 0$ as $t\to\infty$；

(3) $F(t)$一阶可导。

其中 e_0 和 $\dot{e}_0$ 为 $t=0$ 时的位置误差及其导数。条件(1)使系统状态位于滑模面，条件(2)保证了闭环系统稳定，条件(3)是滑模存在条件的要求。

根据上述分析，将 $F(t)$ 定义为

$$F(t)=s(0)\exp(-\lambda t) \tag{4.16}$$

其中 $\lambda>0$，$s(0)$为初始时刻的 $s(t)$。

设计滑模控制律为

$$u=\frac{1}{b}[-f(\theta)+\ddot{r}+c\dot{e}+K(t)\mathrm{sgn}(s)-\dot{F}(t)] \tag{4.17}$$

稳定性证明：

定义 Lyapunov 函数为

$$V=\frac{1}{2}s^2 \tag{4.18}$$

则

$$\begin{aligned}\dot{V}&=s\dot{s}=s[\ddot{e}+c\dot{e}-\dot{F}(t)]=s[\ddot{r}-\ddot{\theta}+c\dot{e}-\dot{F}(t)]\\&=s[\ddot{r}-f(\theta)-bu-bE(t)+c\dot{e}-\dot{F}(t)]\end{aligned} \tag{4.19}$$

将控制律式(4.17)代入式(4.19)，得

$$\dot{V}=s[-K(t)\mathrm{sgn}(s)-E(t)] \tag{4.20}$$

则

$$\dot{V}=-K(t)|s|-E(t)s \tag{4.21}$$

$$\dot{V}\leqslant-\eta|s| \tag{4.22}$$

在滑模控制律式(4.17)中，切换增益 $K(t)$值是造成抖振的原因。$K(t)$用于补偿不确定项 $E(t)$，以保证滑模存在性条件得到满足。如果 $E(t)$时变，则为了降低抖振，$K(t)$也应该时变。

4.3.3 模糊控制器设计

滑模存在条件为

$$s\dot{s}<0 \tag{4.23}$$

二维平面内的滑模运动如图 4-15 所示。

由图 4-15 可见，当系统到达滑模面后，将会保持在滑模面上。$K(t)$为保证系统运动得以到达滑模面的增益，其值必须足以消除不确定项的影响。

模糊规则如下：

如果 $s\dot{s}>0$，则 $K(t)$应增大　(4.24)

如果 $s\dot{s}<0$，则 $K(t)$应减小　(4.25)

由式(4.24)和式(4.25)可设计关于 $s\dot{s}$ 和 $\Delta K(t)$之间关系的模糊系统，在该系统中，$s\dot{s}$ 为输入，$\Delta K(t)$为输出。系统输入输出的模糊集分别定义如下：

$$s\dot{s}=\{\text{NB}\quad \text{NM}\quad \text{ZO}\quad \text{PM}\quad \text{PB}\} \tag{4.26}$$

$$\Delta K=\{\text{NB}\quad \text{NM}\quad \text{ZO}\quad \text{PM}\quad \text{PB}\} \tag{4.27}$$

其中 NB 为负大，NM 为负中，ZO 为零，PM 为正中，PB 为正大。

模糊系统的输入输出隶属函数如图 4-16 和图 4-17 所示。

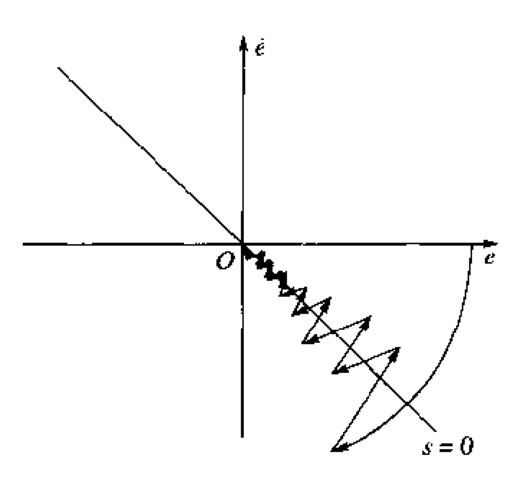


图 4-15　二维平面内的滑模运动

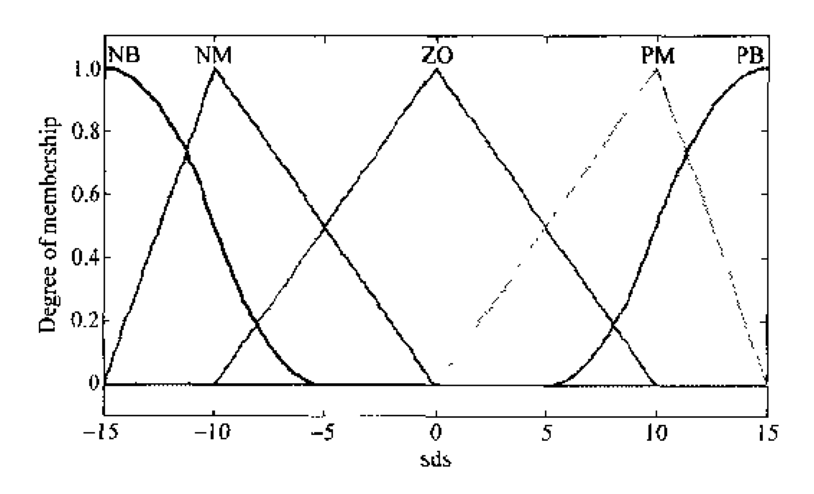


图 4-16　模糊输入的隶属函数

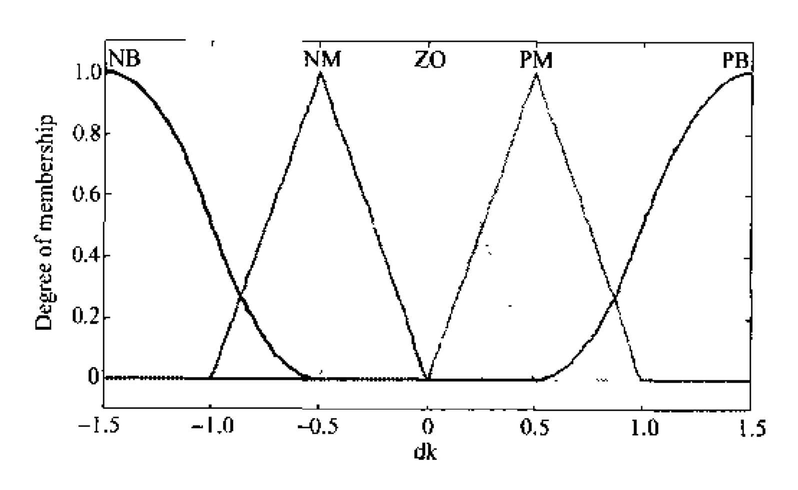


图 4-17　模糊输出的隶属函数

选择如下模糊规则：

$$\left.\begin{aligned}&\text{R1: IF } s\dot{s} \text{ is PB THEN } \Delta K \text{ is PB}\\&\text{R2: IF } s\dot{s} \text{ is PM THEN } \Delta K \text{ is PM}\\&\text{R3: IF } s\dot{s} \text{ is ZO THEN } \Delta K \text{ is ZO}\\&\text{R4: IF } s\dot{s} \text{ is NM THEN } \Delta K \text{ is NM}\\&\text{R5: IF } s\dot{s} \text{ is NB THEN } \Delta K \text{ is NB}\end{aligned}\right\} \tag{4.28}$$

采用积分的方法对$\hat{K}(t)$的上界进行估计：

$$\hat{K}(t)=G\int_0^t \Delta K \mathrm{d}t \tag{4.29}$$

其中 G 为比例系数，$G>0$。

控制系统的结构如图 4-18 所示。

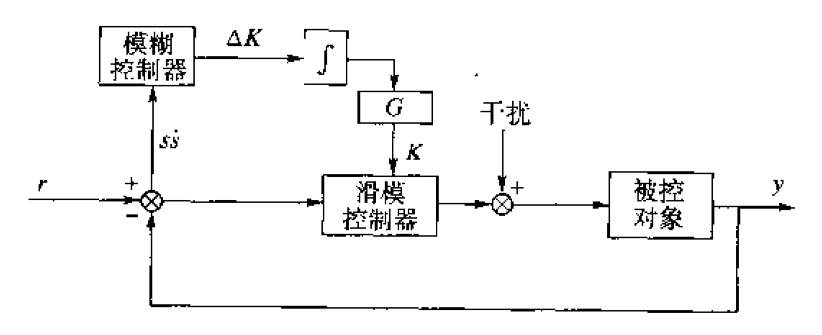


图 4-18 模糊滑模控制系统结构

用$\hat{K}(t)$代替式(4.17)的 $K(t)$，则控制律变为

$$u=\frac{1}{b}(-f(\theta)+\ddot{r}+c\dot{e}+\hat{K}(t)\mathrm{sgn}(s)-\dot{F}(t)) \tag{4.30}$$

4.3.4 仿真实例

考虑如下伺服系统：

$$\ddot{\theta}=f(\theta)+b(u(t)+E(t)) \tag{4.31}$$

假设 $f(\theta)=-25\dot{\theta}$，$b=133$。

1. 实例 1

首先考虑不确定项 $E(t)$为高斯函数的形式：

$$E(t)=200\exp\left(-\frac{(t-c_i)^2}{2b_i^2}\right)$$

取 $b_i=0.50$，$c_i=5.0$，$\eta=1.0$，则控制器的切换增益为 $K(t)=\max(|E(t)|)+\eta=201$。$E(t)$如图 4-19 所示，该函数的程序设计如下。

Gaussian 函数设计程序：chap4_3func.m

```
clear all;
close all;

b=0.5;
```

```
c = 5;
ts = 0.001;
for k = 1:1:10000
  t(k) = k * ts;
  E(k) = 200 * exp( - (t(k) - c)^2/(2 * b^2));
end
figure(1);
plot(t,E);
xlabel('time(s)');
ylabel('Gaussian function');
```

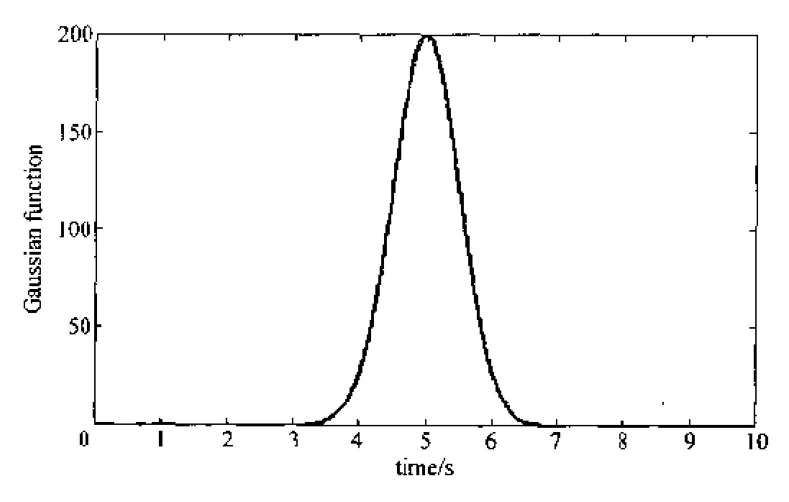


图 4-19 高斯函数形式的不确定项 $E(t)$

位置指令信号为 $r=A\sin(2\pi Ft)$，$A=1.0$，$F=1.0\text{Hz}$。

采用 S 函数程序 chap4_3rule.m 建立模糊系统，其中取 flag=1 时，可给出隶属函数图，如图 4-16 和图 4-17 所示。

首先，取 $M=2$，采用控制律式(4.30)，取 $G=400$，$c=150$，$\lambda=10$。仿真结果如图 4-20～图 4-22 所示。取 $M=1$，采用传统的控制律式(4.17)，取 $D=200$，$c=150$，仿真结果如图 4-23 和图 4-24 所示。

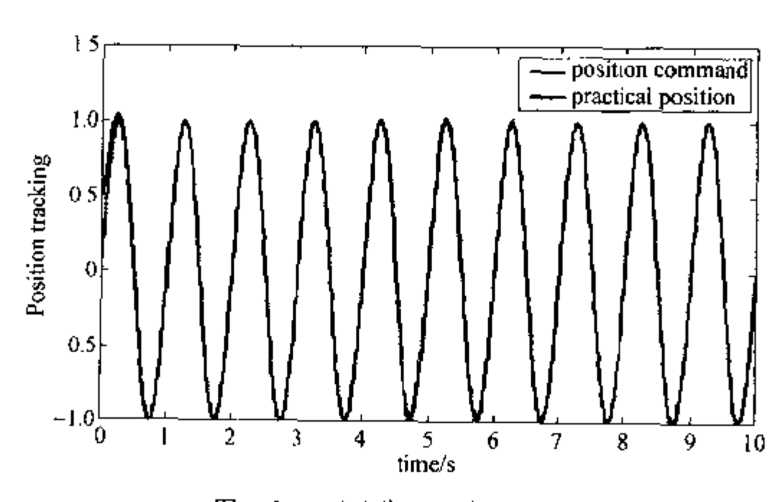


图 4-20 正弦位置跟踪($M=2$)

可见，采用基于模糊规则的模糊滑模控制方法，可有效地通过切换增益消除干扰项，从

而消除抖振。

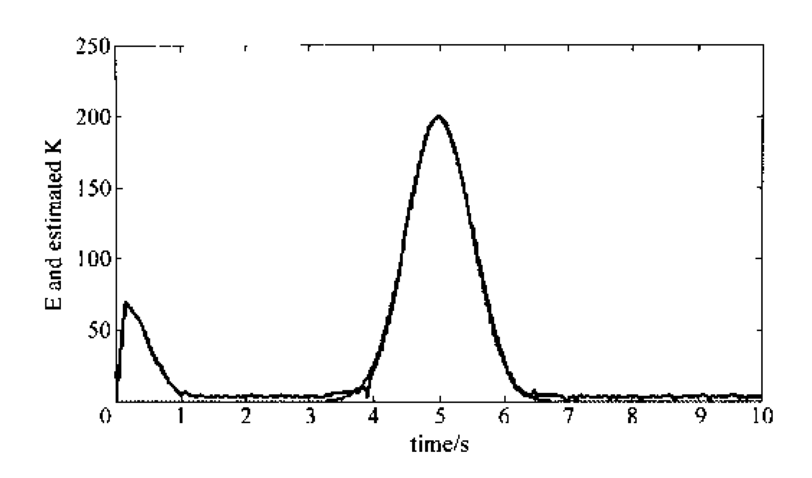


图 4-21 $E(t)$ 及其估计值 $\hat{K}(t)$ $(M=2)$

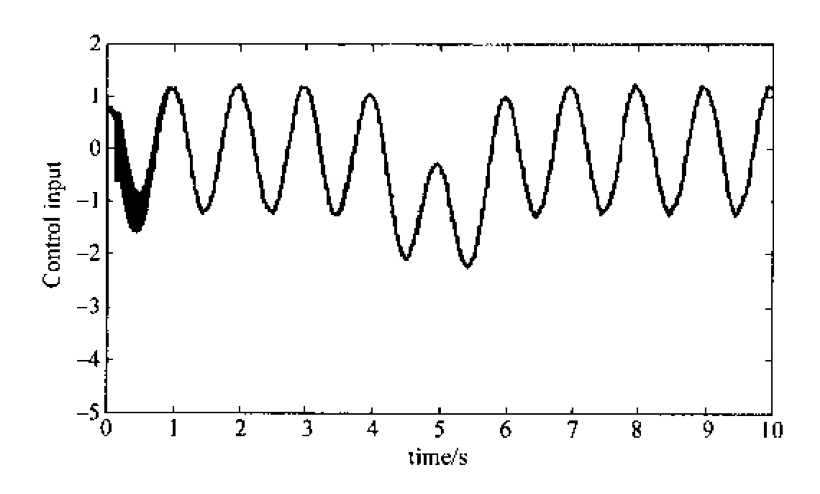


图 4-22 控制输入信号 $(M=2)$

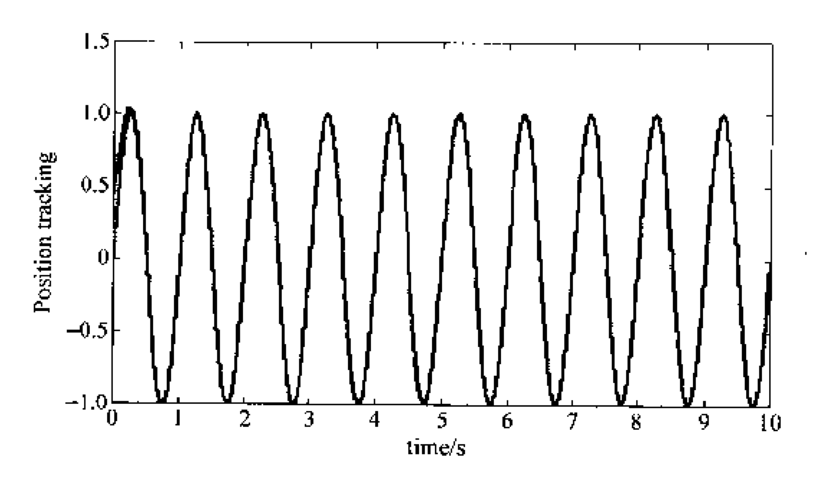


图 4-23 正弦位置跟踪 $(M=1)$

仿真程序如下。

(1) Simulink 主程序(如图 4-25 所示):chap4_3sim.mdl

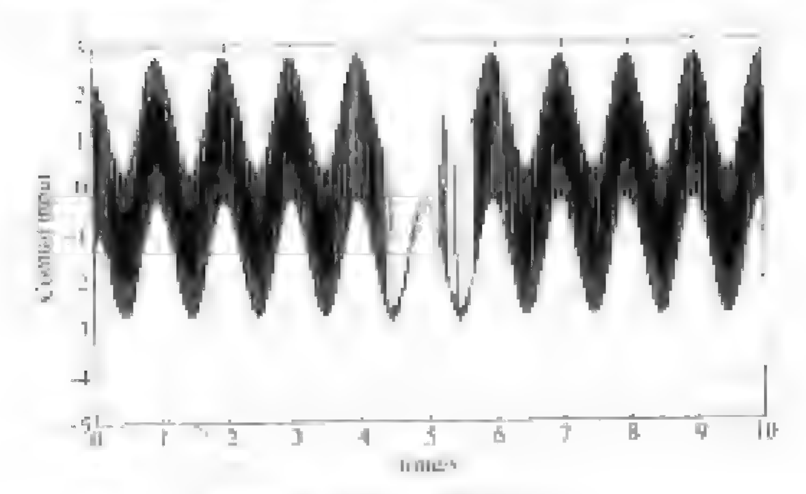


图6-24 控制输入信号(M=1)

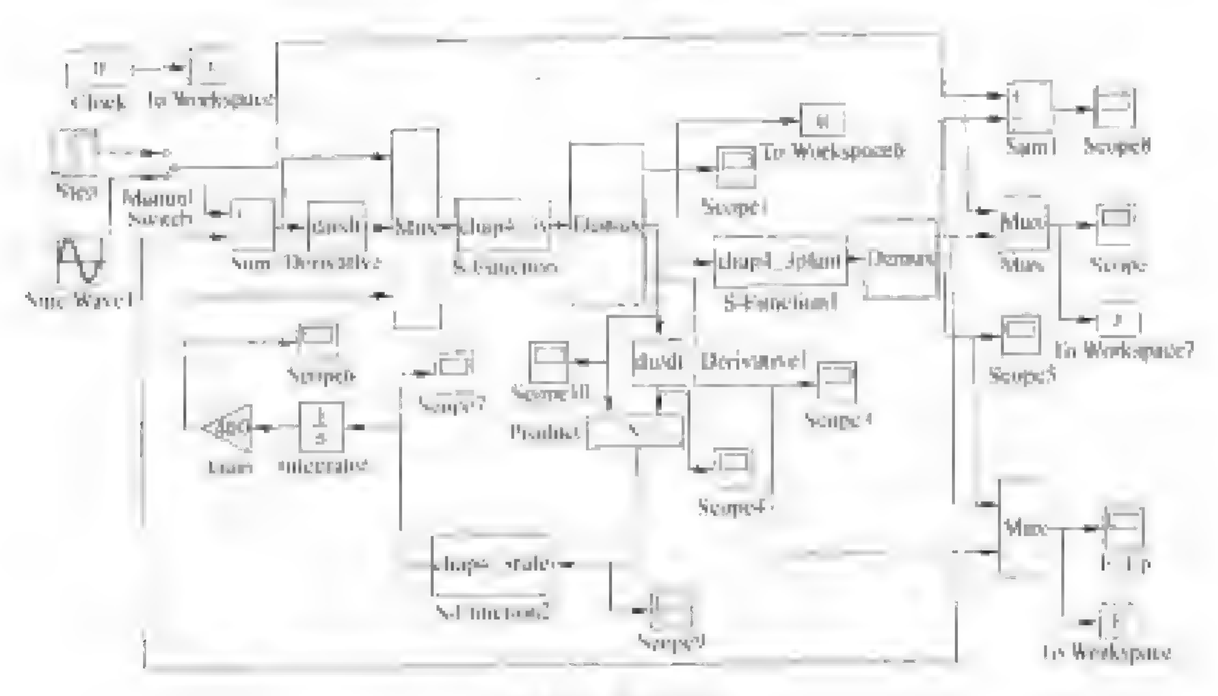


图6-25 [illegible]程序图

(2) 控制器S函数程序：chap4_[illegible]s.m

```
%S-function for continuous state equation
function [sys,x0,str,ts]=s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
%Outputs
  case 3,
    sys=mdlOutputs(t,x,u);
%Unhandled flags
  case {2,4,9}
    sys=[];
```

```
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)
persistent s0

e = u(1);
de = u(2);

c = 150;
if t == 0
  e0 = e;de0 = de;
  s0 = c * e0 + de0;
end

r = sin(2 * pi * t);
dr = 2 * pi * cos(2 * pi * t);
ddr = - (2 * pi)^2 * sin(2 * pi * t);
% r = 1.0;dr = 0;ddr = 0;

x1 = r - e;
x2 = dr - de;

fx = - 25 * x2;
b = 133;

nmn = 10;
Ft = s0 * exp( - nmn * t);
dFt = - s0 * nmn * exp( - nmn * t);

s = c * e + de - Ft;

D = 200;
xite = 1.0;
```

```
M=2;
if M==1
  K=D+xite;
elseif M==2 % Estimation for K with fuzzy
  K=abs(u(3))+xite;
end

ut=1/b*(-fx+ddr+c*de+K*sign(s)-dFt);

sys(1)=ut;
sys(2)=s;
sys(3)=K;
```

(3) 被控对象 S 函数程序:chap4_3plant.m

```
%S-function for continuous state equation
function [sys,x0,str,ts]=s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys=mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs    = 2;
sizes.NumInputs     = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys=simsizes(sizes);
x0=[0.5,0.5];
str=[];
ts=[];
```

```
function sys = mdlDerivatives(t,x,u)

%bi = 0.05;ci = 5;
bi = 0.5;ci = 5;
dt = 200 * exp( - (t - ci)^2/(2 * bi^2)); %rbf_func.m
%dt = 0;

sys(1) = x(2);
sys(2) = - 25 * x(2) + 133 * u + dt;
function sys = mdlOutputs(t,x,u)
%bi = 0.05;ci = 5;
bi = 0.5;ci = 5;
dt = 200 * exp( - (t - ci)^2/(2 * bi^2)); %rbf_func.m
%dt = 0;

sys(1) = x(1);
sys(2) = dt;
```

(4) 模糊系统 S 函数程序:chap4_3rule.m

```
%S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [];
str = [];
```

```
ts = [];

function sys = mdlOutputs(t,x,u)
persistent a1

if t == 0
a = newfis('smc_fuzz');

f1 = 5; % 500/3;
a = addvar(a,'input','sds',[-3*f1,3*f1]);  % Parameter e
a = addmf(a,'input',1,'NB','zmf',[-3*f1,-1*f1]);
a = addmf(a,'input',1,'NM','trimf',[-3*f1,-2*f1,0]);
a = addmf(a,'input',1,'Z','trimf',[-2*f1,0,2*f1]);
a = addmf(a,'input',1,'PM','trimf',[0,2*f1,3*f1]);
a = addmf(a,'input',1,'PB','smf',[1*f1,3*f1]);

f2 = 0.5; % 0.5; % 200/3;
a = addvar(a,'output','dk',[-3*f2,3*f2]);  % Parameter u
a = addmf(a,'output',1,'NB','zmf',[-3*f2,-1*f2]);
a = addmf(a,'output',1,'NM','trimf',[-2*f2,-1*f2,0]);
a = addmf(a,'output',1,'Z','trimf',[-1*f2,0,1*f2]);
a = addmf(a,'output',1,'PM','trimf',[0,1*f2,2*f2]);
a = addmf(a,'output',1,'PB','smf',[1*f2,3*f2]);

rulelist = [1 1 1 1;  % Edit rule base
          2 2 1 1;
          3 3 1 1;
          4 4 1 1;
          5 5 1 1];

a1 = addrule(a,rulelist);
a1 = setfis(a1,'DefuzzMethod','centroid');  % Defuzzy
writefis(a1,'smc_fuzz');

a1 = readfis('smc_fuzz');

flag = 1;
if flag == 1
figure(1);
plotmf(a1,'input',1);
figure(2);
plotmf(a1,'output',1);
end
end

sys(1) = evalfis([u(1)],a1);
```

(5) 作图程序：chap4_3plot.m

```
close all;
```

```
figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,E(:,1),'r',t,E(:,2),'b');
xlabel('time(s)');ylabel('E and estimated K');

figure(3);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');
```

2. 实例 2

在高精度、超低速伺服系统中,由于非线性摩擦环节的存在,使系统的动态及静态性能受到很大程度的影响,这主要表现为低速时出现爬行现象,稳态时有较大的静差或出现极限环振荡。为了减轻机械伺服系统中摩擦环节带来的负面影响,可以采用适当的控制补偿方法,对摩擦力(矩)进行补偿。

目前,已提出的摩擦模型很多,主要有 Karnopp 模型、LuGre 模型及综合模型。其中,LuGre 模型是 Canudas 等在 1995 年提出的典型伺服系统的摩擦模型[3],该模型能够准确地描述摩擦过程的复杂的动态、静态特性,如爬行(stick slip)、极限环振荡(hunting)、滑前变形(presliding displacement)、摩擦记忆(friction memory)、变静摩擦(rising static friction)及静态 Stribeck 曲线。

假设 $E(t)$ 为 LuGre 摩擦模型,该模型描述如下:

$$E(t)=\sigma_0 z+\sigma_1\dot{z}+\alpha\dot{\theta} \tag{4.32}$$

$$\dot{z}=\dot{\theta}-\frac{\sigma_0|\dot{\theta}|}{g(\dot{\theta})}z \tag{4.33}$$

$$g(\dot{\theta})=F_c+(F_s-F_c)\exp(-(\dot{\theta}/V_s)^2)+\alpha\dot{\theta} \tag{4.34}$$

其中 σ_0,σ_1 称为动态摩擦参数,α 为粘性摩擦系数,F_c 为库仑摩擦力,F_s 为静摩擦力,V_s 为切换速度。

取位置指令为 $r=A\sin(2\pi Ft)$,$A=1.0$,$F=0.50\text{Hz}$。在 LuGre 摩擦模型中,取 $\sigma_0=260$,$\sigma_1=2.5$,$\alpha=0.02$,$F_c=280$,$F_s=340$,$V_s=0.01$。通过 Lugre 摩擦模型,可得不确定性的上界,则 $K=\max(|E(t)|)+\eta=250$。

采用 S 函数程序 chap4_4rule.m 建立模糊系统,其中取 flag=1 时,可给出隶属函数图,如图 4-16 和图 4-17 所示。

取 $M=2$,采用控制律式(4.30),取 $G=5000$,$c=150$,$\lambda=10$。仿真结果如图 4-26~图 4-28 所示。取 $M=1$,采用传统控制律式(4.17),取 $D=200$,$c=31$,仿真结果如图 4-29 和图 4-30 所示。可见,采用模糊滑模控制,可有效估计出不确定性的上界,从而消除抖振。

仿真程序如下。

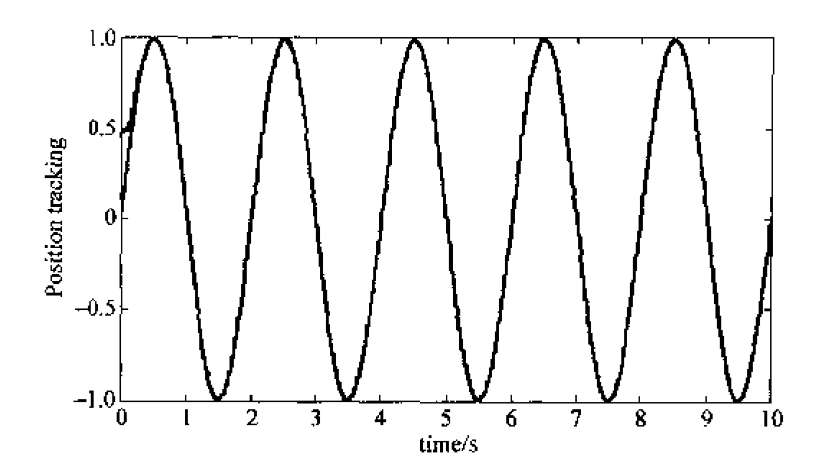


图 4-26　采用 FGSMC 时的位置跟踪（$M=2$）

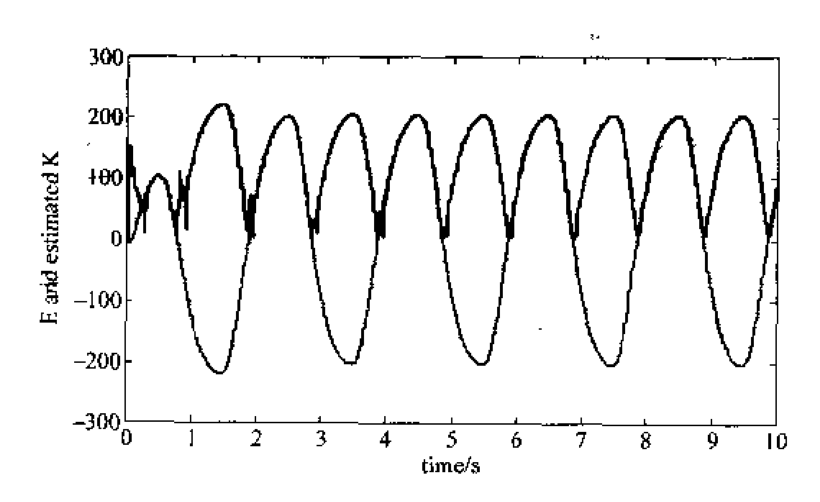


图 4-27　实际摩擦力及其估计值（$M=2$）

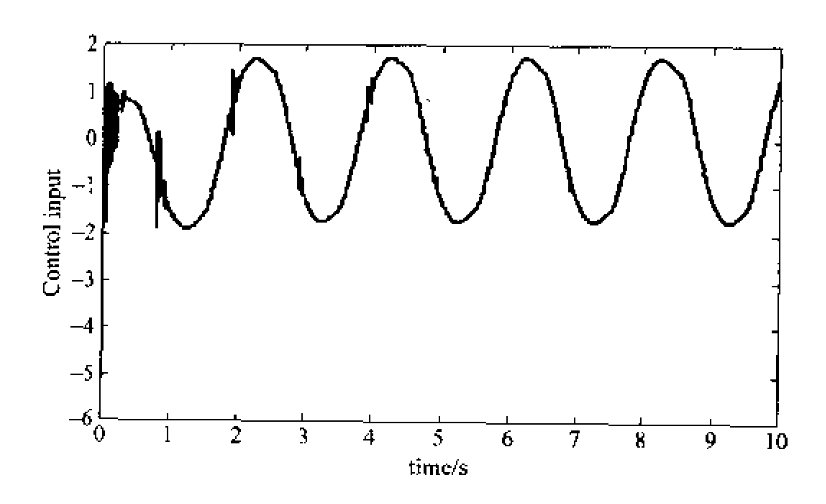


图 4-28　采用 FGSMC 时的控制输入信号（$M=2$）

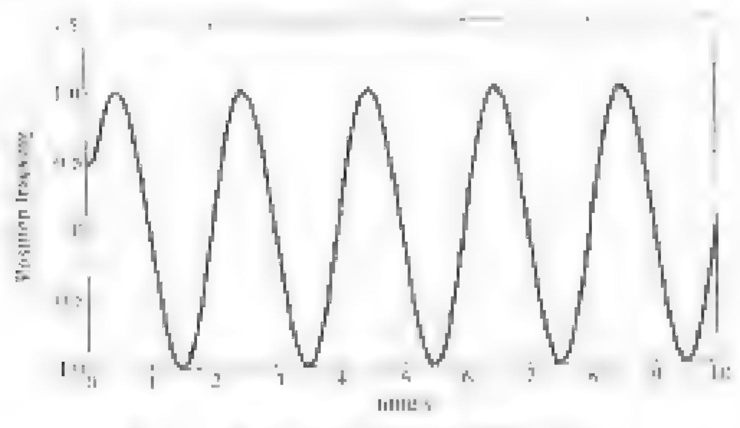


图4-29　采用GSMC时的位置跟踪（M=1）

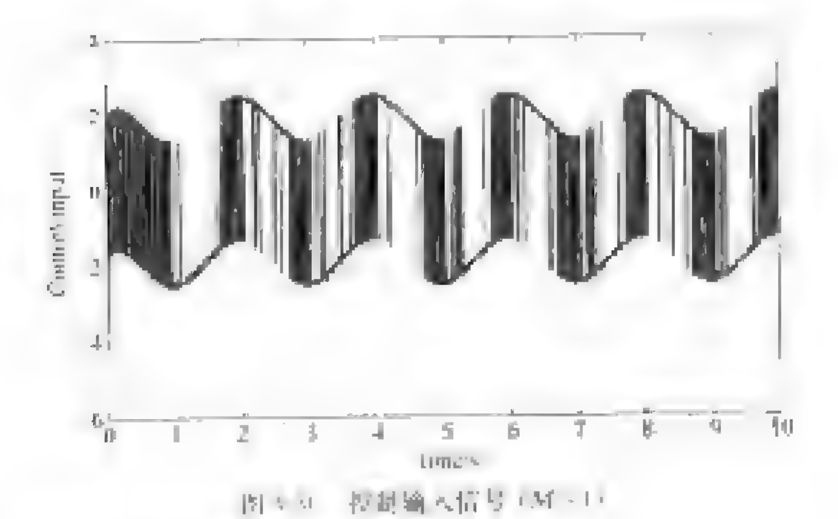


图4-30　控制输入信号（M=1）

（1）Simulink主程序（如图4-31所示）：chap4_4sim.mdl

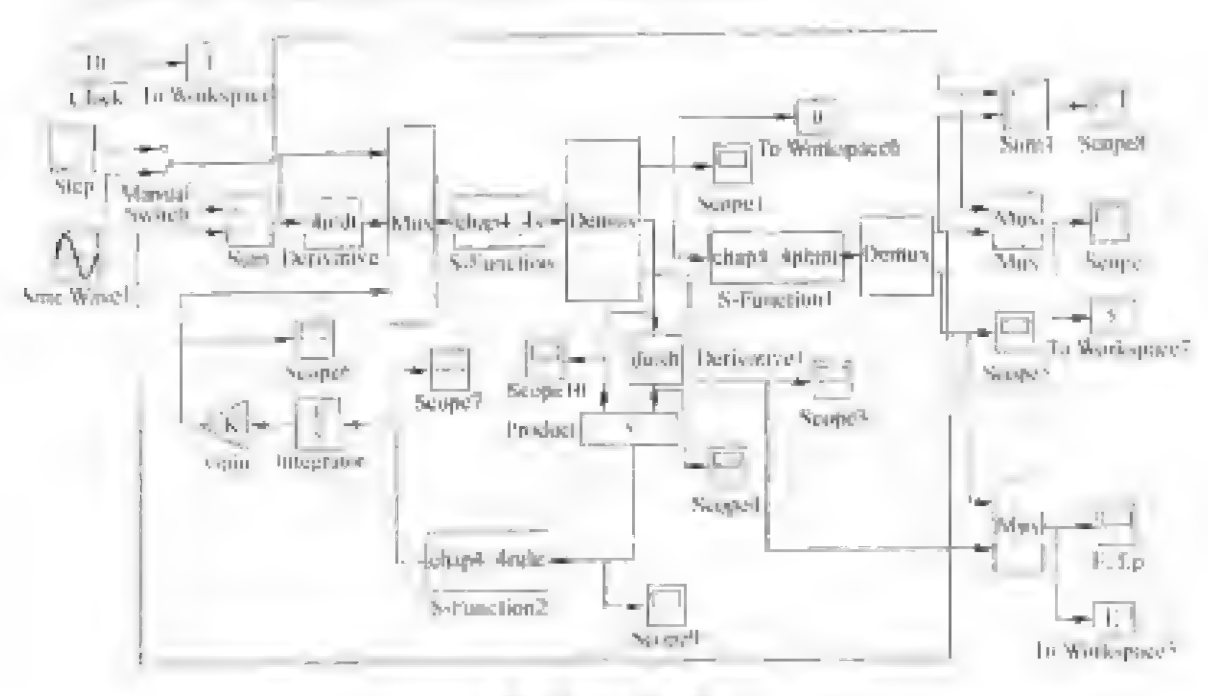


图4-31　主程序图

(2) 控制器S函数程序:chap4_4s.m

```
%S-function for continuous state equation
function [sys,x0,str,ts]=s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
%Outputs
  case 3,
    sys=mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs  = 3;
sizes.NumInputs  = 3;
sizes.DirFeedthrough  = 1;
sizes.NumSampleTimes  = 0;

sys=simsizes(sizes);
x0=[];
str=[];
ts=[];

function sys=mdlOutputs(t,x,u)
persistent s0

e=u(1);
de=u(2);

c=150;
if t==0
   e0=e;de0=de;
     s0=c*e0+de0;
end

r=1*sin(0.5*2*pi*t);
dr=1*2*pi*0.5*cos(0.5*2*pi*t);
ddr=-1*(2*pi*0.5)^2*sin(0.5*2*pi*t);
%r=1.0;dr=0;ddr=0;
```

```
x1 = r - e;
x2 = dr - de;

fx = - 25 * x2;
b = 133;

nmn = 10;
Ft = s0 * exp( - nmn * t);
dFt = - s0 * nmn * exp( - nmn * t);

s = c * e + de - Ft;

D = 250;
xite = 1.0;

M = 1;
if M == 1
  K = D + xite;
elseif M == 2 % Estimation for K with fuzzy
  K = abs(u(3)) + xite;
end

ut = 1/b * ( - fx + ddr + c * de + K * sign(s) - dFt);
% ut = 50 * e + 0.01 * de;

sys(1) = ut;
sys(2) = s;
sys(3) = K;
```

(3) 被控对象 S 函数程序：chap4_4plant.m

```
% S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
```

```
%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 3;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0.5,0.5,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)

%Friction
    Fc = 0.28 * 1000;
    Fs = 0.34 * 1000;
    vs = 0.01;
    af = 0.02;
    rou0 = 260;
    rou1 = 2.5;
    g = Fc + (Fs - Fc) * exp( - (x(2)/vs)^2) + af * x(2);
    sys(3) = x(2) - (rou0 * abs(x(2))/g) * x(3);  % ?????
    dt = rou0 * x(3) + rou1 * sys(3) + af * x(2);

sys(1) = x(2);
sys(2) = -25 * x(2) + 133 * u - dt;
function sys = mdlOutputs(t,x,u)

%Friction
    Fc = 0.28 * 1000;
    Fs = 0.34 * 1000;
    vs = 0.01;
    af = 0.02;
    rou0 = 260;
    rou1 = 2.5;
    g = Fc + (Fs - Fc) * exp( - (x(2)/vs)^2) + af * x(2);
    Q = x(2) - (rou0 * abs(x(2))/g) * x(3);  % ?????
    dt = rou0 * x(3) + rou1 * Q + af * x(2);

sys(1) = x(1);
sys(2) = dt;
```

(4) 模糊系统 S 函数程序：chap4_4rule.m

```
%S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)
```

```
switch flag,
 % Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
 % Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
 % Unhandled flags
  case {2, 4, 9 }
    sys = [];
 % Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

 % mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs  = 1;
sizes.NumInputs  = 1;
sizes.DirFeedthrough  = 1;
sizes.NumSampleTimes  = 0;

sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)
persistent a1

if t == 0
a = newfis('smc_fuzz');

f1 = 5; % 500/3;
a = addvar(a,'input','sds',[-3*f1,3*f1]);  % Parameter e
a = addmf(a,'input',1,'NB','zmf',[-3*f1,-1*f1]);
a = addmf(a,'input',1,'NM','trimf',[-3*f1,-2*f1,0]);
a = addmf(a,'input',1,'Z','trimf',[-2*f1,0,2*f1]);
a = addmf(a,'input',1,'PM','trimf',[0,2*f1,3*f1]);
a = addmf(a,'input',1,'PB','smf',[1*f1,3*f1]);

f2 = 0.5; % 0.5; % 200/3;
a = addvar(a,'output','dk',[-3*f2,3*f2]);  % Parameter u
a = addmf(a,'output',1,'NB','zmf',[-3*f2,-1*f2]);
a = addmf(a,'output',1,'NM','trimf',[-2*f2,-1*f2,0]);
a = addmf(a,'output',1,'Z','trimf',[-1*f2,0,1*f2]);
a = addmf(a,'output',1,'PM','trimf',[0,1*f2,2*f2]);
a = addmf(a,'output',1,'PB','smf',[1*f2,3*f2]);
```

```
rulelist=[1 1 1 1;  % Edit rule base
          2 2 1 1;
          3 3 1 1;
          4 4 1 1;
          5 5 1 1];

a1 = addrule(a,rulelist);
a1 = setfis(a1,'DefuzzMethod','centroid');  % Defuzzy
writefis(a1,'smc_fuzz');

a1 = readfis('smc_fuzz');

flag = 1;
if flag == 1
figure(1);
plotmf(a1,'input',1);
figure(2);
plotmf(a1,'output',1);
end
end

sys(1) = evalfis([u(1)],a1);
```

(5) 作图程序:同 chap4_3plot.m。

4.4　基于等效控制的模糊滑模控制

滑模控制器中,控制律通常由等效控制 u_{eq} 和切换控制 u_{sw} 组成。等效控制将系统状态保持在滑模面上,切换控制迫使系统状态在滑模面上滑动。利用模糊规则,可建立基于等效控制和切换控制的模糊系统,从而消除抖振[3]。

4.4.1　系统描述

考虑 SISO 的 n 阶非线性系统

$$x^{(n)}=f(\boldsymbol{x},t)+g(\boldsymbol{x},t)u(t)+d(t) \tag{4.35}$$

$$\boldsymbol{x}=[x,\dot{x},\cdots,x^{(n-1)}]^{\mathrm{T}},y=x \tag{4.36}$$

其中 $\boldsymbol{x}\in\mathbf{R}^n,u\in\mathbf{R},y\in\mathbf{R}$。假设 $|d(t)|\leqslant D$。

4.4.2　滑模控制器设计

1. 等效滑模控制器的设计

设被控对象为

$$x^{(n)}=f(\boldsymbol{x},t)+g(\boldsymbol{x},t)u(t) \tag{4.37}$$

系统跟踪误差为

$$e=x_d-x=[e\dot{e}\cdots e^{(n-1)}]^T \tag{4.38}$$

则切换函数为

$$s(x,t)=Ce=c_1e+c_2\dot{e}+\cdots+e^{(n-1)} \tag{4.39}$$

其中 $C=[c_1c_2\cdots c_{n-1}1]$

通过取 $\dot{s}=0$，可得

$$\dot{s}(x,t)=c_1\dot{e}+c_2\ddot{e}+\cdots+e^{(n)}=c_1\dot{e}+c_2\ddot{e}+\cdots+c_{n-1}e^{(n-1)}+x_d^{(n)}-x^{(n)}$$

$$=\sum_{i=1}^{n-1}c_ie^{(i)}+x_d^{(n)}-f(x,t)-g(x,t)u(t)=0 \tag{4.40}$$

则等效控制器为

$$u_{eq}=\frac{1}{g(x,t)}\left(\sum_{i=1}^{n-1}c_ie^{(i)}+x_d^{(n)}-f(x,t)\right) \tag{4.41}$$

2. 滑模控制器的设计

为了满足滑模到达条件 $s(x,t)\cdot\dot{s}(x,t)\leqslant-\eta|s|$，其中 $\eta>0$，必须采用切换控制。切换控制器设计为

$$u_{sw}=\frac{1}{g(x,t)}\eta\text{sgn}(s) \tag{4.42}$$

其中 $\eta\geqslant D$。

滑模控制器为

$$u=u_{eq}+u_{sw} \tag{4.43}$$

稳定性证明：

$$\dot{s}(x,t)=\sum_{i=1}^{n-1}c_ie^{(i)}+x_d^{(n)}-f(x,t)-g(x,t)u(t)-d(t) \tag{4.44}$$

将式(4.41)~式(4.43)代入式(4.44)，得

$$s\dot{s}=s\cdot(-\eta\cdot\text{sgn}(s))-s\cdot d(t)=-\eta|s|-sd(t)\leqslant0 \tag{4.45}$$

4.4.3 模糊控制器设计

根据滑模控制的原理，如果滑模控制器由等效滑模控制和切换控制两部分构成，其控制规则为

$$\text{If } s(t) \text{ is ZO then } u \text{ is } u_{eq} \tag{4.46}$$

$$\text{If } s(t) \text{ is NZ then } u \text{ is } u_{eq}+u_{sw} \tag{4.47}$$

其中模糊集 ZO 和 NZ 分别表示“零”和“非零”。

模糊规则式(4.46)表示当切换函数 $s(t)$ 为零时，模糊控制器为等效控制 u_{eq}。模糊规则式(4.47)表示当切换函数 $s(t)$ 为非零时，模糊控制器为切换控制 u_{sw}。

采用反模糊化方法，模糊控制器设计为

$$u=\frac{\mu_{ZO}(s)u_{eq}+\mu_{NZ}(s)(u_{eq}+u_{sw})}{\mu_{ZO}(s)+\mu_{NZ}(s)}=u_{eq}+\mu_{NZ}(s)u_{sw} \tag{4.48}$$

$$\mu_{ZO}(s)+\mu_{NZ}(s)=1 \tag{4.49}$$

当 $\mu_{NZ}(s)=1$ 时，$u=u_{eq}+u_{sw}$，此时控制律为传统的等效滑模控制。当 $\mu_{NZ}(s)\neq1$ 时，通过隶属函数 $\mu_{NZ}(s)$ 的变化实现抖振的消除。

4.4.4 仿真实例

考虑一个实际伺服系统

$$\ddot{x}=-25\dot{x}+133u(t)+d(t) \tag{4.50}$$

假设 $f(\boldsymbol{x},t)=-25\dot{x}$，$b(\boldsymbol{x},t)=133$。

考虑如下高斯函数形式的干扰 $d(t)$：

$$d(t)=200\exp\left(-\frac{(t-c_i)^2}{2b_i^2}\right) \tag{4.51}$$

取 $b_i=0.50$，$c_i=5.0$，$\eta=5.0$。则干扰上界为 $D=\max(|d(t)|)+\eta=205$。干扰 $d(t)$ 如图 4-32 所示。输入位置指令为 $r=A\sin(2\pi Ft)$，其中 $A=1.0$，$F=2.0\text{Hz}$。

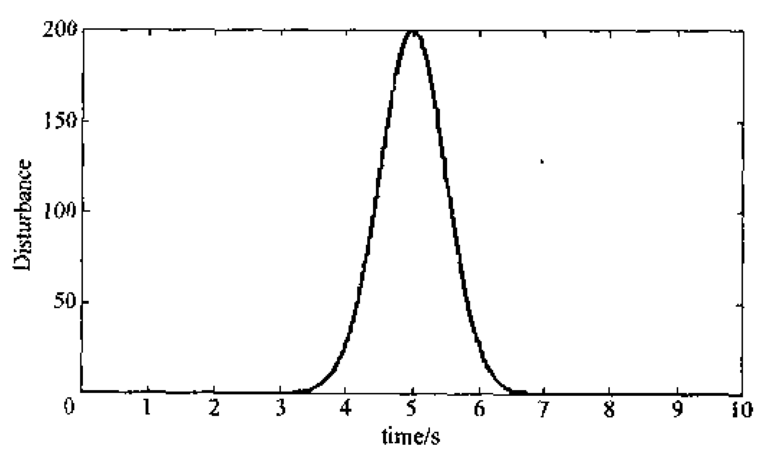


图 4-32 高斯函数形式的干扰

在 S 函数程序 chap4_5s.m 中建立模糊系统，并通过命令 persistent 将规则库一直保持在运行过程中。取 $F=1$，可给出隶属函数图。模糊系统输入和输出的隶属函数形式如图 4-33和图 4-34 所示。模糊规则设计为

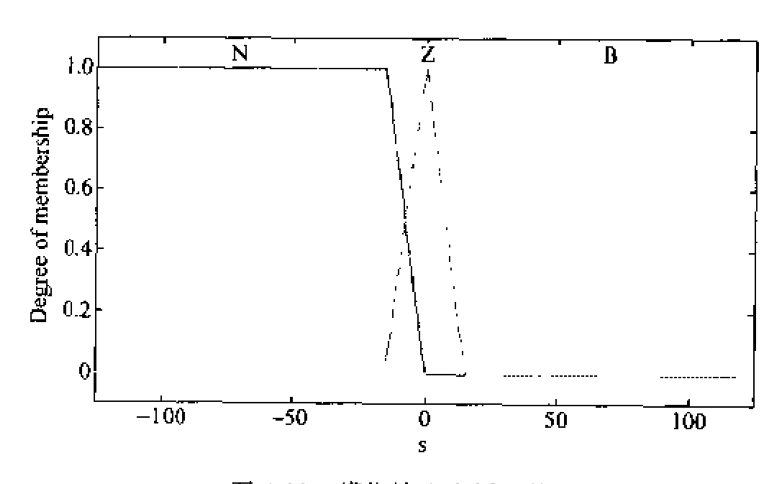


图 4-33 模糊输入隶属函数

(1) If (s is N) then (u is B)

(2) If (s is Z) then (u is Z)

(3) If (s is B) then (u is B)

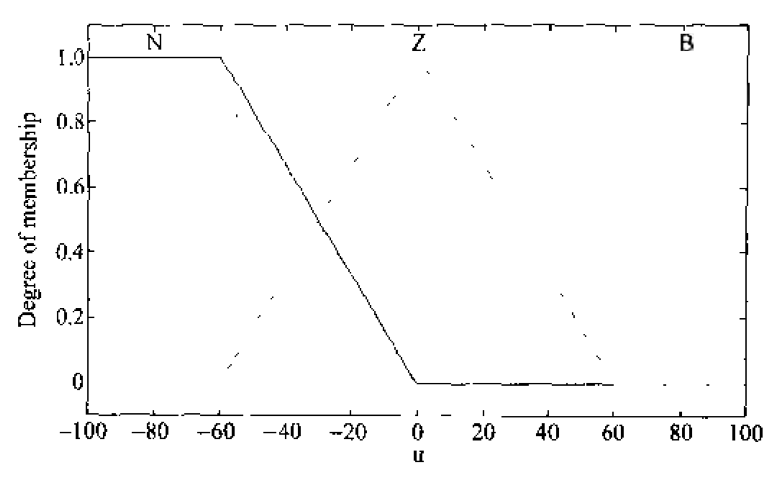


图 4-34 模糊输出隶属函数

首先,取 $M=1$,采用传统的等效滑模控制律式(4.43),取 $c=25$,仿真结果如图 4-35～图 4-37 所示。取 $M=2$,采用控制律式(4.48),取 $c=25$,仿真结果如图 4-38～图 4-41 所示。可见,采用基于等效控制的模糊滑模控制,可有效地消除抖振。

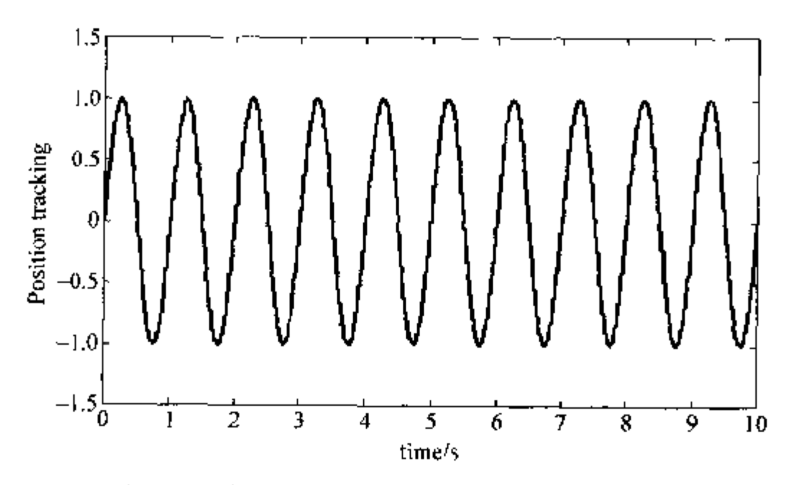


图 4-35 传统等效滑模控制正弦位置跟踪($M=1$)

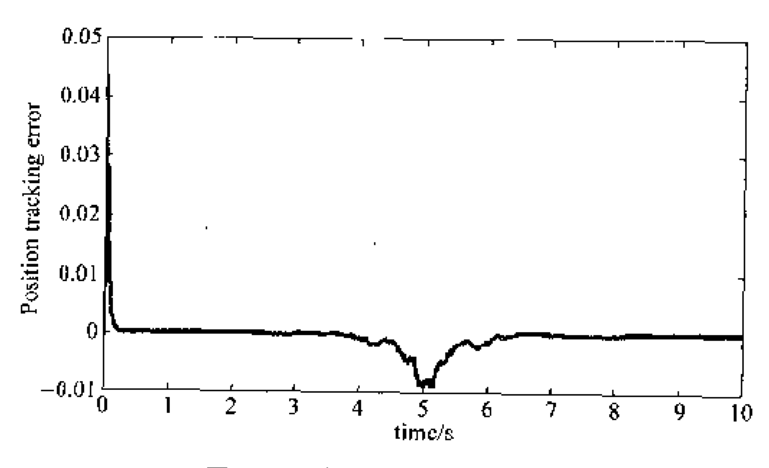


图 4-36 位置跟踪误差($M=1$)

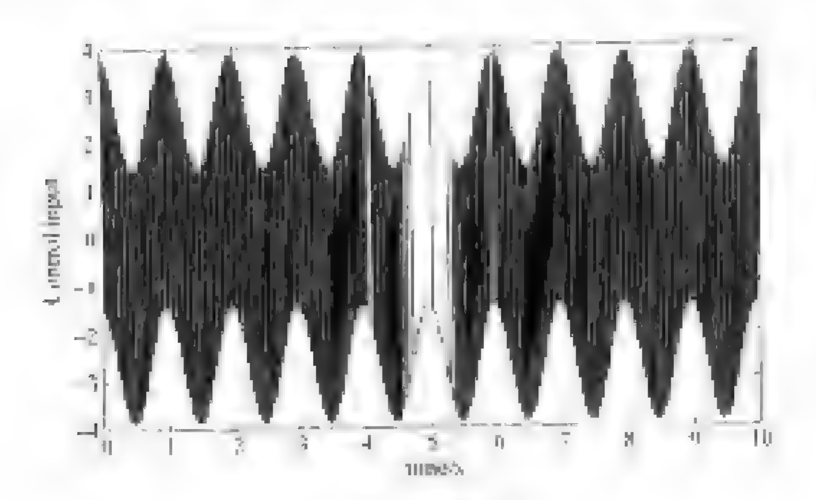


图4-37　控制输入信号(M=1)

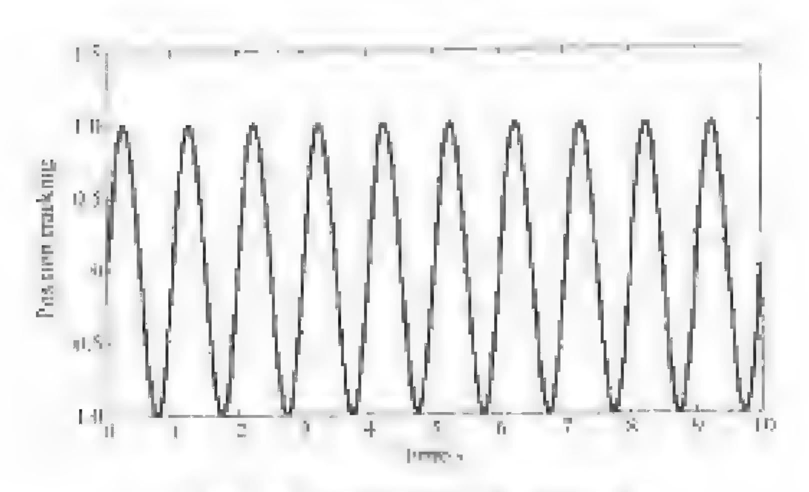


图4-38　模糊滑模控制位置跟踪(M=2)

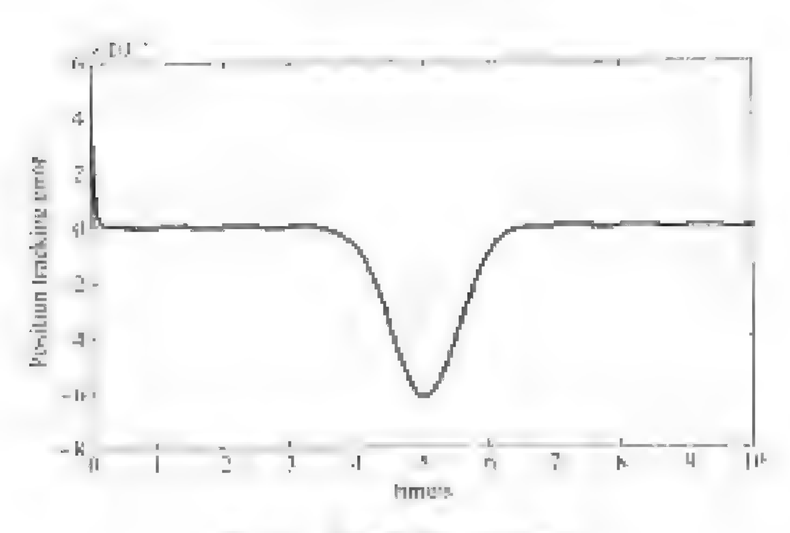


图4-39　位置跟踪误差(M=2)

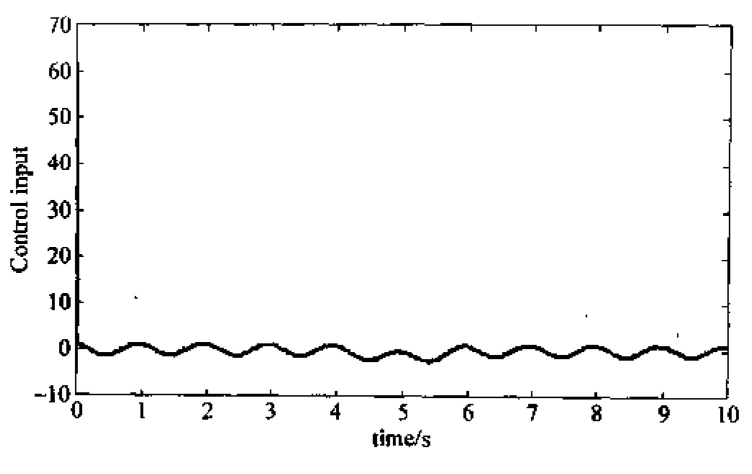


图 4-40 控制输入信号($M=2$)

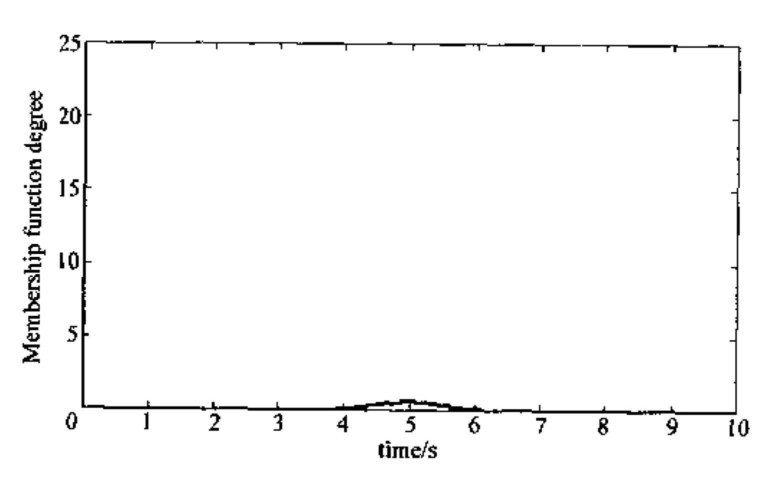


图 4-41 隶属函数 $\mu_{NZ}(s)$的变化($M=2$)

仿真程序如下。

(1) Simulink 主程序(如图 4-42 所示):chap4_5sim.mdl

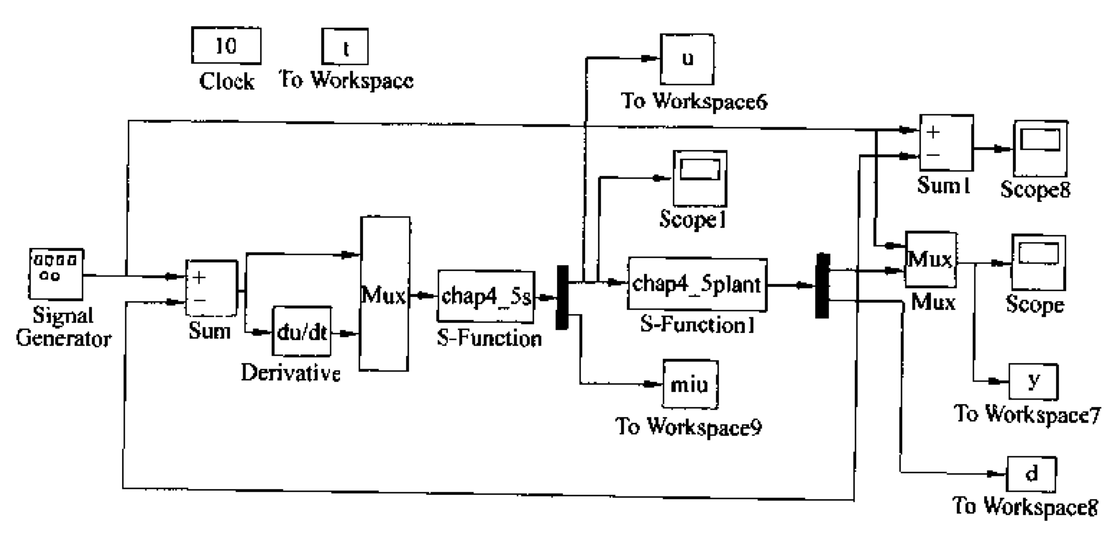


图 4-42 主程序图

(2) 控制器 S 函数:chap4_5s.m

```
%S-function for continuous state equation
function [sys,x0,str,ts]=s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
%Outputs
  case 3,
    sys=mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys=simsizes(sizes);
x0=[];
str=[];
ts=[];

function sys=mdlOutputs(t,x,u)
persistent a2

if t==0
a=newfis('fuzz_smc');

a=addvar(a,'input','s',5*[-25,25]);
a=addmf(a,'input',1,'N','trapmf',5*[-25,-25,-3,0]);
a=addmf(a,'input',1,'Z','trimf',5*[-3,0,3]);
a=addmf(a,'input',1,'B','trapmf',5*[0,3,25,25]);

a=addvar(a,'output','u',20*[-5,5]);
a=addmf(a,'output',1,'N','trapmf',20*[-5,-5,-3,0]);
a=addmf(a,'output',1,'Z','trimf',20*[-3,0,3]);
a=addmf(a,'output',1,'B','trapmf',20*[0,3,5,5]);

rulelist=[1 3 1 1;
        2 2 1 1;
```

```
        3 3 1 1];

a = addrule(a,rulelist);
%showrule(a)                    % Show fuzzy rule base

a1 = setfis(a,'DefuzzMethod','centroid');  % Defuzzy
writefis(a1,'fsmc');            % Save fuzzy system as "fsmc.fis"
a2 = readfis('fsmc');

F = 1;
  if F == 1
    figure(1);
    plotmf(a,'input',1);
    figure(2);
    plotmf(a,'output',1);
  end
end

r = sin(2*pi*t);
dr = 2*pi*cos(2*pi*t);
ddr = -(2*pi)^2*sin(2*pi*t);

e = u(1);
de = u(2);

c = 25;
s = c*e + de;

x2 = dr - de;
f = -25*x2;
b = 133;

ueq = 1/b*(c*de + ddr - f);
D = 200;
xite = D + 150;
usw = 1/b*xite*sign(s);

M = 2;
if M == 1
  miu = 1.0;
elseif M == 2
  miu = evalfis([s],a2);  % Using fuzzy inference
end
ut = ueq + miu*usw;

sys(1) = ut;
sys(2) = miu;
```

(3) 被控对象 S 函数：chap4_5plant.m

```
%S-function for continuous state equation
```

```
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
 % Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
  sys = mdlDerivatives(t,x,u);
 % Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
 % Unhandled flags
  case {2, 4, 9 }
    sys = [];
 % Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

 % mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)

bi = 0.50;ci = 5;
dt = 200 * exp( - (t - ci)^2/(2 * bi^2)); % rbf_func.m

sys(1) = x(2);
sys(2) = - 25 * x(2) + 133 * u + dt;
function sys = mdlOutputs(t,x,u)
bi = 0.50;ci = 5;
dt = 200 * exp( - (t - ci)^2/(2 * bi^2)); % rbf_func.m

sys(1) = x(1);
sys(2) = dt;
```

(4) 作图程序:chap4_5plot.m

```
close all;
```

```
figure(1);
plot(t,d(:,1),'r');
xlabel('time(s)');ylabel('Disturbance');

figure(2);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(3);
plot(t,y(:,1) - y(:,2),'r');
xlabel('time(s)');ylabel('Position tracking error');

figure(4);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(5);
plot(t,miu(:,1),'r');
xlabel('time(s)');ylabel('Membership function degree');
```

4.5 基于线性化反馈的自适应模糊滑模控制

利用线性化反馈方法，可设计滑模控制器。采用模糊逼近及自适应控制方法，利用线性化反馈技术，可设计一种自适应模糊滑模控制器[5]。

4.5.1 线性化反馈方法

考虑如下 SISO 系统：

$$\begin{aligned}\dot{x}&=f_0+g_0(x)u\\ y&=h(x)\end{aligned} \tag{4.52}$$

其中 $x\in\mathbf{R}^n$ 为状态变量，$f_0,g_0:\mathbf{R}^n\to\mathbf{R}^n$，$h:\mathbf{R}^n\to\mathbf{R}^n$，且 $f_0(0)=0,h(0)=0$。则

$$\begin{aligned}\dot{y}&=\frac{\partial h}{\partial x}\dot{x}=\frac{\partial h}{\partial x}f_0(x)+\frac{\partial h}{\partial x}g_0(x)u\\ &\overset{\text{def}}{=}f_1(x)+g_1(x)u\end{aligned} \tag{4.53}$$

假设 $g_1(x)\neq 0$，可设计如下线性化反馈控制律：

$$u=\frac{R-f_1(x)}{g_1(x)} \tag{4.54}$$

则式(4.53)变为线性系统

$$\dot{y}=R \tag{4.55}$$

设位置指令为 $y_d(t)$，取 R 为

$$R=\dot{y}_d-\alpha(y-y_d) \tag{4.56}$$

其中 $\alpha>0$。则式(4.56)又变为

$$\dot{e}+\alpha e=0 \tag{4.57}$$

其中 $e=y-y_d$。

显然，式(4.57)为误差动态方程，$e(t)$以指数形式趋近于零。如果 $e(0)=\dot{e}(0)=0$，则 $e(t)$在所有时间($t\geqslant 0$)都为零。

4.5.2　滑模控制器设计

考虑如下 n 阶 SISO 非线性系统：

$$x^{(n)}=f(\boldsymbol{x},t)+g(\boldsymbol{x},t)u \tag{4.58}$$

其中 f 和 g 为未知非线性函数，$x\in\mathbf{R}^n$ 为状态变量。

设位置指令为 $\boldsymbol{x}_d$，则跟踪误差为

$$\boldsymbol{e}=\boldsymbol{x}-\boldsymbol{x}_d=[e\dot{e}\cdots e^{(n-1)}]^{\mathrm{T}} \tag{4.59}$$

定义滑模面为

$$s(\boldsymbol{x},t)=\boldsymbol{ce} \tag{4.60}$$

其中 $\boldsymbol{c}=[c_1c_2\cdots c_{n-1}1]$。

根据线性化反馈技术，将滑模控制律设计为

$$u=\frac{R-f(x,t)}{g(x,t)} \tag{4.61}$$

$$\xi_1(\boldsymbol{x},t)=\ddot{\boldsymbol{x}}_d-\boldsymbol{c}_1\dot{\boldsymbol{e}} \tag{4.62}$$

$$R=\xi_1(\boldsymbol{x},t)-k\mathrm{sgn}(s),k>0 \tag{4.63}$$

稳定性证明：

定义 Lyapunov 函数

$$V=\frac{1}{2}s^2 \tag{4.64}$$

针对二阶系统，有 $s=c_1e+\dot{e}$，则

$$\begin{aligned}\dot{V}&=s\dot{s}=s(\ddot{e}+\boldsymbol{c}_1\dot{\boldsymbol{e}})=s(\ddot{x}-\ddot{x}_d+\boldsymbol{c}_1\dot{\boldsymbol{e}})\\&=s(f(\boldsymbol{x},t)+g(\boldsymbol{x},t)u-\ddot{x}_d+\boldsymbol{c}_1\dot{\boldsymbol{e}})\end{aligned} \tag{4.65}$$

将式(4.61)代入式(4.65)得

$$\dot{V}=s(-k\mathrm{sgn}(s)) \tag{4.66}$$

即

$$\dot{V}=-k|s| \tag{4.67}$$

$$\dot{V}\leqslant 0 \tag{4.68}$$

4.5.3　自适应模糊滑模控制器设计

如果 $f(\boldsymbol{x},t)$和 $g(\boldsymbol{x},t)$未知，可采用模糊系统$\hat{f}(\boldsymbol{x},t)$和$\hat{g}(\boldsymbol{x},t)$代替 $f(\boldsymbol{x},t)$和 $g(\boldsymbol{x},t)$，实现自适应模糊滑模控制。

1. 基本的模糊系统

设模糊系统由 IF-THEN 形式的模糊规则构成：

$$R^{(j)}: \text{IF } x_1 \text{ is } A_1^j \text{ and } \cdots \text{ and } x_n \text{ is } A_n^j \text{ THEN } y \text{ is } B^j \tag{4.69}$$

采用乘积推理机、单值模糊器和中心平均解模糊器，则模糊系统的输出为

$$y(\boldsymbol{x}) = \frac{\sum_{j=1}^{m} y^j \left(\prod_{i=1}^{n} \mu_{A_i^j}(x_i) \right)}{\sum_{j=1}^{m} \left(\prod_{i=1}^{n} \mu_{A_i^j}(x_i) \right)} \tag{4.70}$$

其中 $\mu_{A_i^j}(x_i)$ 为 x_i 的隶属函数。

引入向量 $\boldsymbol{\xi}(x)$，式(4.70)变为

$$y(\boldsymbol{x}) = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{x}) \tag{4.71}$$

其中 $\boldsymbol{\theta} = [y^1 \cdots y^m]^{\mathrm{T}}$，$\boldsymbol{\xi}(\boldsymbol{x}) = [\xi^1(\boldsymbol{x}) \cdots \xi^m(\boldsymbol{x})]^{\mathrm{T}}$。

$$\boldsymbol{\xi}(x) = \frac{\prod_{i=1}^{n} \mu_{A_i^j}(x_i)}{\sum_{j=1}^{m} \left(\prod_{i=1}^{n} \mu_{A_i^j}(x_i) \right)} \tag{4.72}$$

2. 自适应模糊滑模控制器的设计

在实际控制中，f 和 g 往往未知，控制律式(4.61)很难实现。采用模糊系统逼近 f 和 g，则控制律式(4.61)变为

$$u(t) = \frac{R - \hat{f}(\boldsymbol{x},t)}{\hat{g}(\boldsymbol{x},t)} \tag{4.73}$$

$$\hat{f}(\boldsymbol{x}|\boldsymbol{\theta}_f) = \boldsymbol{\theta}_f^{\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{x}), \quad \hat{g}(\boldsymbol{x}|\boldsymbol{\theta}_g) = \boldsymbol{\theta}_g^{\mathrm{T}} \boldsymbol{\xi}(\boldsymbol{x}) \tag{4.74}$$

$\boldsymbol{\xi}(x)$ 为模糊向量，参数 $\boldsymbol{\theta}_f^{\mathrm{T}}$ 和 $\boldsymbol{\theta}_g^{\mathrm{T}}$ 根据自适应律而变化。设计自适应律为

$$\dot{\boldsymbol{\theta}}_f = -r_1 s\, \boldsymbol{\xi}(\boldsymbol{x}) \tag{4.75}$$

$$\dot{\boldsymbol{\theta}}_g = -r_2 s\, \boldsymbol{\xi}(\boldsymbol{x}) u \tag{4.76}$$

稳定性证明：

设最优参数为

$$\boldsymbol{\theta}_f^* = \arg \min_{\boldsymbol{\theta}_f \in \Omega_f} \left[\sup_{x \in \mathbf{R}^n} \left| \hat{f}\left(\boldsymbol{x} | \boldsymbol{\theta}_f \right) - f(\boldsymbol{x},t) \right| \right] \tag{4.77}$$

$$\boldsymbol{\theta}_g^* = \arg \min_{\boldsymbol{\theta}_g \in \Omega_g} \left[\sup_{x \in \mathbf{R}^n} \left| \hat{g}\left(\boldsymbol{x} | \boldsymbol{\theta}_g \right) - g(\boldsymbol{x},t) \right| \right] \tag{4.78}$$

其中 Ω_f 和 Ω_g 分别为 $\boldsymbol{\theta}_f$ 和 $\boldsymbol{\theta}_g$ 的集合。

定义最小逼近误差为

$$w = f(\boldsymbol{x},t) - \hat{f}(\boldsymbol{x}|\boldsymbol{\theta}_f^*) + (g(\boldsymbol{x},t) - \hat{g}(x|\boldsymbol{\theta}_g^*))u \tag{4.79}$$

针对二阶系统，$n=2$，则

$$\begin{aligned} \dot{s} &= \boldsymbol{c}_1 \dot{\boldsymbol{e}} + \ddot{\boldsymbol{e}} = \boldsymbol{c}_1 \dot{\boldsymbol{e}} + \ddot{x} - \ddot{x}_d = c_1 \dot{\boldsymbol{e}} + f(x,t) + g(\boldsymbol{x},t)u - \ddot{x}_d \\ &= f(\boldsymbol{x},t) + g(\boldsymbol{x},t)u + \boldsymbol{c}_1 \dot{\boldsymbol{e}} - \ddot{x}_d = f(\boldsymbol{x},t) + g(\boldsymbol{x},t)u - \xi(\boldsymbol{x},t) \end{aligned}$$

将控制律 u 代入上式，得

$$\begin{aligned}\dot{s} &= f(\boldsymbol{x},t)+[g(\boldsymbol{x},t)+\hat{g}(\boldsymbol{x},t)-\hat{g}(\boldsymbol{x},t)]u-\xi_1(\boldsymbol{x},t)\\ &= f(\boldsymbol{x},t)+[g(\boldsymbol{x},t)-\hat{g}(\boldsymbol{x},t)]u+\hat{g}(\boldsymbol{x},t)[\hat{g}^{-1}(\boldsymbol{x},t)(-\hat{f}(\boldsymbol{x},t)+R)]-\xi_1(\boldsymbol{x},t)\\ &=[f(\boldsymbol{x},t)-\hat{f}(\boldsymbol{x},t)]+[g(\boldsymbol{x},t)-\hat{g}(\boldsymbol{x},t)]u+R-\xi_1(\boldsymbol{x},t)\\ &=[f(\boldsymbol{x},t)-\hat{f}(\boldsymbol{x},t)]+[g(\boldsymbol{x},t)-\hat{g}(\boldsymbol{x},t)]u+\xi_1(\boldsymbol{x},t)-k\operatorname{sgn}(s)-\xi_1(\boldsymbol{x},t)\\ &=[f(\boldsymbol{x},t)-\hat{f}(\boldsymbol{x},t)]+[g(\boldsymbol{x},t)-\hat{g}(\boldsymbol{x},t)]u-k\operatorname{sgn}(s)\end{aligned}$$

由式(4.79)得

$$\begin{aligned}\dot{s} &= \hat{f}^*(\boldsymbol{x},t)-\hat{f}(\boldsymbol{x},t)+(\hat{g}^*(\boldsymbol{x},t)-\hat{g}(\boldsymbol{x},t))u-k\operatorname{sgn}(s)+w\\ &=\boldsymbol{\varphi}_f^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})+\boldsymbol{\varphi}_g^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})u(t)-k\operatorname{sgn}(s)+w\end{aligned} \tag{4.80}$$

其中 $\boldsymbol{\varphi}_f=\boldsymbol{\theta}_f-\boldsymbol{\theta}_f^*$，$\boldsymbol{\varphi}_g=\boldsymbol{\theta}_g-\boldsymbol{\theta}_g^*$，$\hat{f}^*(\boldsymbol{x},t)=\hat{f}(\boldsymbol{x}|\boldsymbol{\theta}_f^*)$，$\hat{g}^*(\boldsymbol{x},t)=\hat{g}^*(\boldsymbol{x}|\boldsymbol{\theta}_g^*)$。

定义 Lyapunov 函数

$$V=\frac{1}{2}\left(s^2+\frac{1}{r_1}\boldsymbol{\varphi}_f^{\mathrm{T}}\boldsymbol{\varphi}_f+\frac{1}{r_2}\boldsymbol{\varphi}_g^{\mathrm{T}}\boldsymbol{\varphi}_g\right) \tag{4.81}$$

其中 r_1 和 r_2 为正常数。

则

$$\begin{aligned}\dot{V} &= s\dot{s}+\frac{1}{r_1}\boldsymbol{\varphi}_f^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_f+\frac{1}{r_2}\boldsymbol{\varphi}_g^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_g\\ &=s(\boldsymbol{\varphi}_f^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})+\boldsymbol{\varphi}_g^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})u(t)-k\operatorname{sgn}(s)+w)+\frac{1}{r_1}\boldsymbol{\varphi}_f^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_f+\frac{1}{r_2}\boldsymbol{\varphi}_g^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_g\\ &=s\boldsymbol{\varphi}_f^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})+\frac{1}{r_1}\boldsymbol{\varphi}_f^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_f+s\boldsymbol{\varphi}_g^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})u(t)+\frac{1}{r_2}\boldsymbol{\varphi}_g^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_g-k|s|+sw\\ &=\frac{1}{r_1}\boldsymbol{\varphi}_f^{\mathrm{T}}(r_1s\,\boldsymbol{\xi}(\boldsymbol{x})+\dot{\boldsymbol{\varphi}}_f)+\frac{1}{r_2}\boldsymbol{\varphi}_g^{\mathrm{T}}(r_2s\,\boldsymbol{\xi}(\boldsymbol{x})u(t)+\dot{\boldsymbol{\varphi}}_g)-k|s|+sw\end{aligned} \tag{4.82}$$

其中 $\dot{\boldsymbol{\varphi}}_f=\dot{\boldsymbol{\theta}}_f$，$\dot{\boldsymbol{\varphi}}_g=\dot{\boldsymbol{\theta}}_g$。

将式(4.75)和式(4.76)代入式(4.82)，得

$$\dot{V}=-k|s|+sw \tag{4.83}$$

根据模糊逼近理论，自适应模糊系统可实现使逼近误差 w 非常小。因此

$$\dot{V}\leqslant 0$$

为了降低抖振，采用连续函数 S_δ 代替 $\operatorname{sgn}(s)$：

$$S_\delta=\frac{s}{|s|+\delta} \tag{4.84}$$

$$\delta=\delta_0+\delta_1\|e\| \tag{4.85}$$

其中 δ_0，δ_1 为两个正常数。

4.5.4 仿真实例

被控对象取单级倒立摆，其动态方程如下：

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=\dfrac{g\sin x_1-mlx_2^2\cos x_1\sin x_1/(m_c+m)}{l(4/3-m\cos^2x_1/(m_c+m))}+\dfrac{\cos x_1/(m_c+m)}{l(4/3-m\cos^2x_1/(m_c+m))}u\end{cases}$$

其中 x_1 和 x_2 分别为摆角和摆速；$g=9.8\mathrm{m/s^2}$；m_c 为小车质量，$m_c=1\mathrm{kg}$；m 为摆杆质量，$m=0.1\mathrm{kg}$；l 为摆长的一半，$l=0.5\mathrm{m}$；u 为控制输入。

位置指令为 $x_d(t)=0.1\sin(\pi t)$，切换函数为 $s=c_1e+\dot{e}$，$c=5$。取5种隶属函数：$\mu_{NM}(x_i)=\exp[-((x_i+\pi/6)/(\pi/24))^2]$，$\mu_{NS}(x_i)=\exp[-((x_i+\pi/12)/(\pi/24))^2]$，$\mu_{ZO}(x_i)=\exp[-(x_i/(\pi/24))^2]$，$\mu_{PS}(x_i)=\exp[-((x_i-\pi/12)/(\pi/24))^2]$，$\mu_{PM}(x_i)=\exp[-((x_i-\pi/6)/(\pi/24))^2]$。则用于逼近 f 和 g 的模糊规则分别有25条。隶属函数设计程序如下。

隶属函数设计程序：chap4_6mf.m

```
clear all;
close all;

L1 = -pi/6;
L2 = pi/6;
L = L2 - L1;

T = L * 1/1000;

x = L1:T:L2;
figure(1);
for i = 1:1:5
   gs = -[(x + pi/6 - (i - 1) * pi/12)/(pi/24)].^2;
   u = exp(gs);
   hold on;
   plot(x,u);
end
xlabel('x');ylabel('Membership function degree');
```

根据隶属函数设计程序，可得到隶属函数图，如图4-43所示。

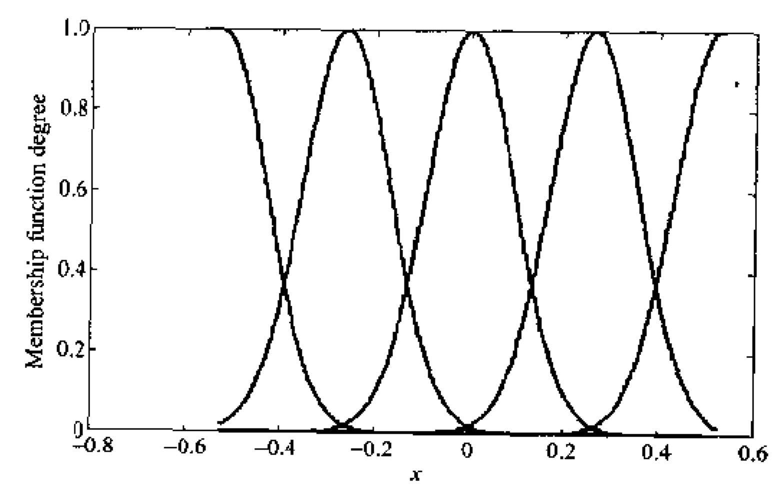


图4-43 x_i 的隶属函数

设 θ_f 和 θ_g 的初始值为0.20，采用控制律式(4.73)，倒立摆初始状态为$[-\pi/600]$。自适应参数取 $r_1=5$，$r_2=1$。

在程序中，分别用fsd，fsu和fs表示模糊系统 $\xi(x)$ 的分子、分母及 $\xi(x)$。$M=1$ 为采用

符号函数，$M=2$ 为采用连续函数式(4.84)。取 $M=2$，$\delta_0=0.03$，$\delta_1=5$，$\delta=\delta_0+\delta_1\|e\|$，$k=5$。仿真结果如图 4-44～图 4-47 所示。

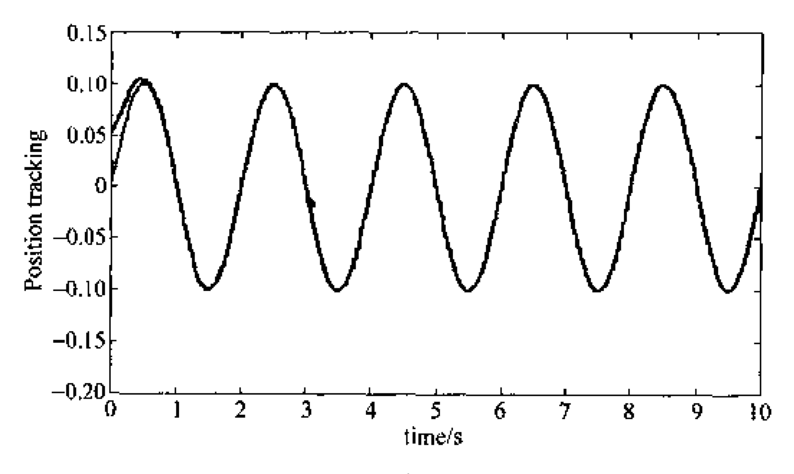


图 4-44　位置跟踪

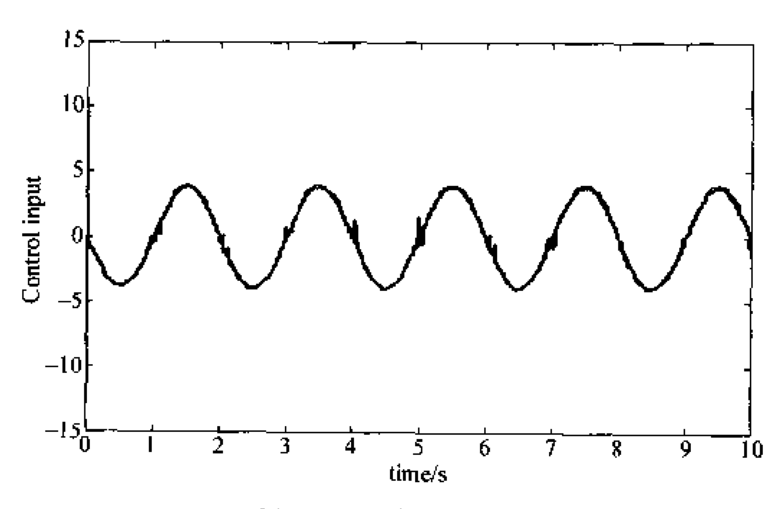


图 4-45　控制输入信号

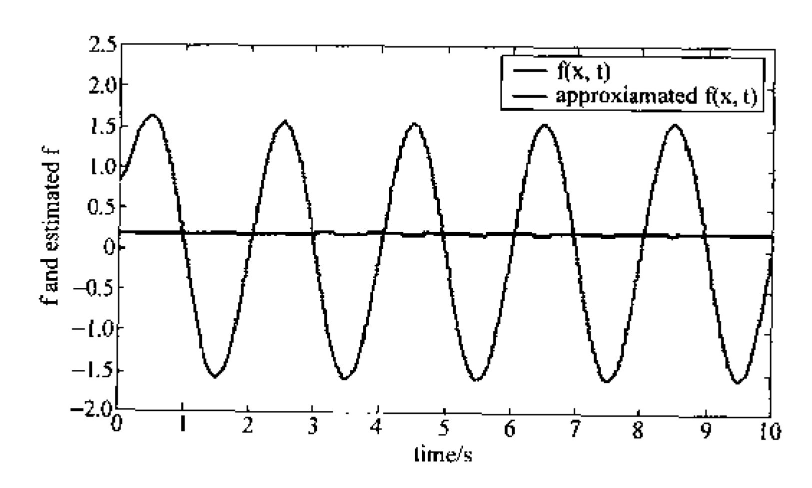


图 4-46　$f(x,t)$ 及 $\hat{f}(x,t)$ 的变化

仿真程序如下。

(1) Simulink 主程序(如图 4-48 所示)：chap4_6sim.mdl

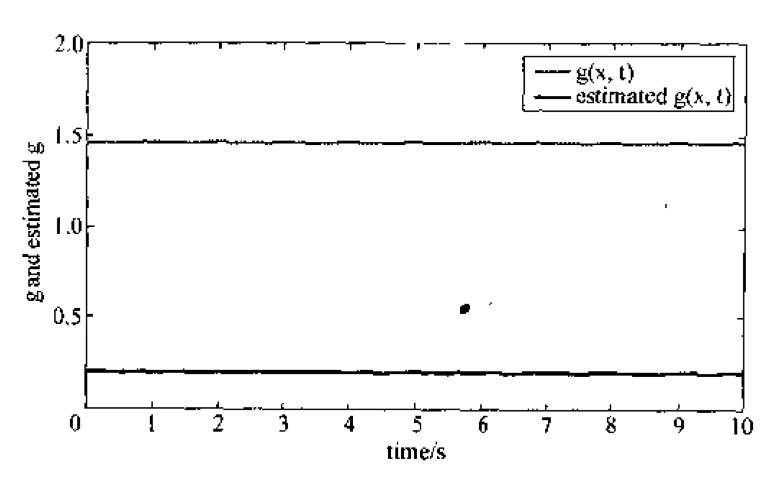


图 4-47　$g(x,t)$及$\hat{g}(x,t)$的变化

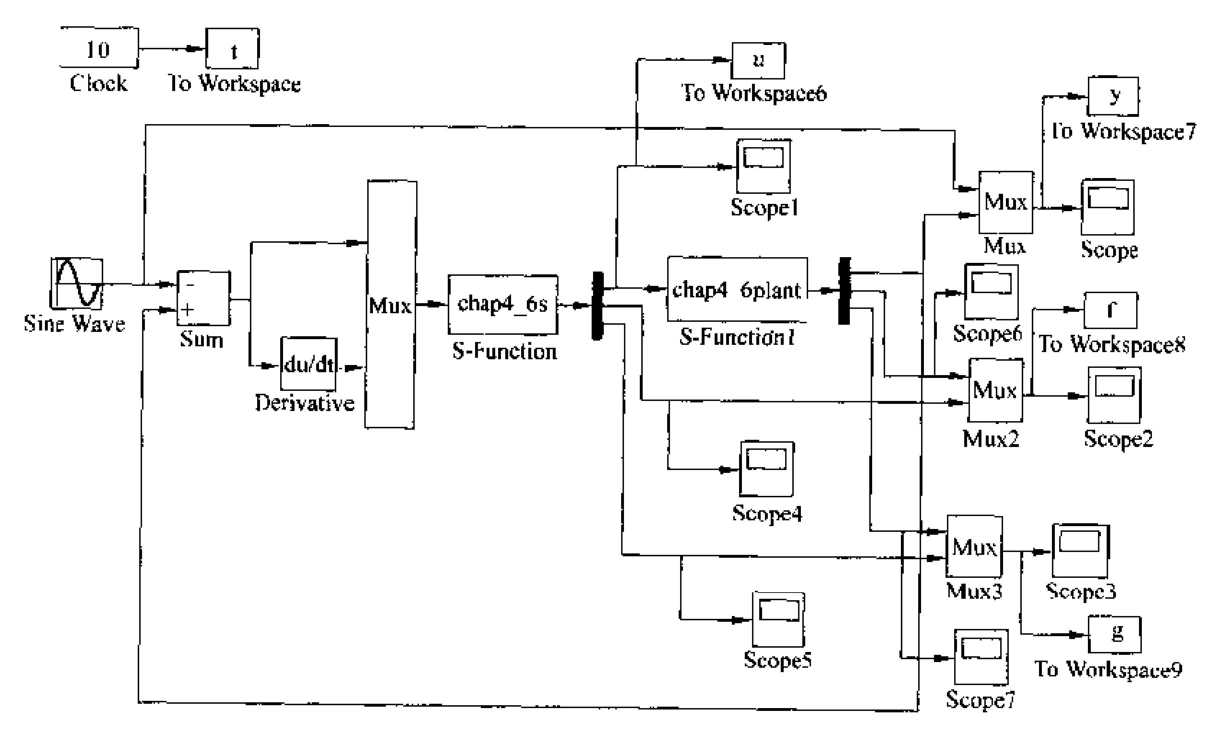


图 4-48　主程序图

(2) 控制器 S 函数:chap4_6s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
```

```
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
global M

sizes = simsizes;
sizes.NumContStates = 50;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0.2 * ones(50,1)];
str = [];
ts = [];

M = 2;
function sys = mdlDerivatives(t,x,u)
global M

r = 0.1 * sin(pi * t);
dr = 0.1 * pi * cos(pi * t);
ddr = - 0.1 * pi * pi * sin(pi * t);

e = u(1);
de = u(2);

xx(1) = r + e;
xx(2) = dr + de;

alfa = 5;
rou = de + alfa * e;
kesi = ddr - alfa * de;

if M == 1
   K = 1.0;
   R = kesi - K * sign(rou);
elseif M == 2
  K = 5;
  delta0 = 0.03;
  delta1 = 5;
  delta = delta0 + delta1 * abs(e);
  R = kesi - K * rou/(abs(rou) + delta);
end

for i = 1:1:25
    thtaf(i,1) = x(i);
end
for i = 1:1:25
```

```
    thtag(i,1) = x(i + 25);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fsd = 0;
for l1 = 1:1:5
  gs1 = -[(xx(1) + pi/6 - (l1 - 1) * pi/12)/(pi/24)]^2;
  u1(l1) = exp(gs1);
end

for l2 = 1:1:5
  gs2 = -[(xx(2) + pi/6 - (l2 - 1) * pi/12)/(pi/24)]^2;
   u2(l2) = exp(gs2);
end

for l1 = 1:1:5
  for l2 = 1:1:5
    fsu(5 * (l1 - 1) + l2) = u1(l1) * u2(l2);
    fsd = fsd + u1(l1) * u2(l2);
  end
end

fs = fsu/(fsd + 0.001);

fx1 = thtaf' * fs';
gx1 = thtag' * fs' + 0.001;

ut = ( - fx1 + R)/gx1;

r1 = 5.0;r2 = 1;

S1 = - r1 * rou * fs;
S2 = - r2 * rou * fs * ut;

for i = 1:1:25
   sys(i) = S1(i);
end
for j = 26:1:50
   sys(j) = S2(j - 25);
end

function sys = mdlOutputs(t,x,u)
global M

r = 0.1 * sin(pi * t);
dr = 0.1 * pi * cos(pi * t);
ddr = - 0.1 * pi * pi * sin(pi * t);

e = u(1);
de = u(2);
```

```
xx(1) = r + e;
xx(2) = dr + de;

alfa = 5;
rou = de + alfa * e;
kesi = ddr - alfa * de;

if M == 1
  K = 1.0;
  R = kesi - K * sign(rou);
elseif M == 2
  K = 5;
  delta0 = 0.03;
  delta1 = 5;
  delta = delta0 + delta1 * abs(e);
  R = kesi - K * rou/(abs(rou) + delta);
end

for i = 1:1:25
   thtaf(i,1) = x(i);
end
for i = 1:1:25
   thtag(i,1) = x(i + 25);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fsd = 0;
for l1 = 1:1:5
   gs1 = -[(xx(1) + pi/6 - (l1 - 1) * pi/12)/(pi/24)]^2;
   u1(l1) = exp(gs1);
end

for l2 = 1:1:5
   gs2 = -[(xx(2) + pi/6 - (l2 - 1) * pi/12)/(pi/24)]^2;
   u2(l2) = exp(gs2);
end

for l1 = 1:1:5
  for l2 = 1:1:5
    fsu(5 * (l1 - 1) + l2) = u1(l1) * u2(l2);
    fsd = fsd + u1(l1) * u2(l2);
  end
end

fs = fsu/(fsd + 0.001);

fx1 = thtaf'* fs';
gx1 = thtag'* fs' + 0.001;

ut = ( - fx1 + R)/gx1;
```

```
sys(1) = ut;
sys(2) = fx1;
sys(3) = gx1;
```

(3) 被控对象 S 函数:chap4_6plant. m

```
% S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [pi/60 0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)

g = 9.8;
mc = 1.0;
m = 0.1;
l = 0.5;

S = l * (4/3 - m * (cos(x(1)))^2/(mc + m));
fx = g * sin(x(1)) - m * l * x(2)^2 * cos(x(1)) * sin(x(1))/(mc + m);
fx = fx/S;
```

```
gx = cos(x(1))/(mc + m);
gx = gx/S;

sys(1) = x(2);
sys(2) = fx + gx * u + 3 * sin(pi * t) * 1;

function sys = mdlOutputs(t,x,u)

g = 9.8;
mc = 1.0;
m = 0.1;
l = 0.5;

S = l * (4/3 - m * (cos(x(1)))^2/(mc + m));
fx = g * sin(x(1)) - m * l * x(2)^2 * cos(x(1)) * sin(x(1))/(mc + m);
fx = fx/S;
gx = cos(x(1))/(mc + m);
gx = gx/S;

sys(1) = x(1);
sys(2) = fx;
sys(3) = gx;
```

(4) 作图程序：chap4_6plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,f(:,1),'r',t,f(:,2),'b');
xlabel('time(s)');ylabel('f and estiamted f');

figure(4);
plot(t,g(:,1),'r',t,g(:,2),'b');
xlabel('time(s)');ylabel('g and estimated g');
```

4.6　基于切换模糊化的自适应模糊滑模控制

利用自适应模糊控制方法，通过将滑模控制器中的切换项进行模糊逼近，可将切换项连续化，从而有效地降低抖振[1]。

4.6.1 系统描述

考虑如下 n 阶 SISO 非线性对象：

$$x^{(n)}=f(\boldsymbol{x},t)+g(\boldsymbol{x},t)u(t)+d(t) \tag{4.86}$$
$$y=x$$

其中 f 和 g 均为未知非线性函数，$x\in\mathbf{R}^n$，$u\in\mathbf{R}$，$y\in\mathbf{R}$，$d(t)$为未知干扰，$|d(t)|\leqslant D$，$g(\boldsymbol{x},t)\neq 0$，$g(\boldsymbol{x},t)>0$。

4.6.2 自适应模糊滑模控制器设计

定义切换函数为

$$s(\boldsymbol{x},t)=-(k_1e+k_2\dot{e}+\cdots+k_{n-1}e^{(n-2)}+e^{(n-1)})=-\boldsymbol{ke} \tag{4.87}$$

其中 $\boldsymbol{e}=\boldsymbol{x}_d-\boldsymbol{x}=[e\dot{e}\cdots e^{(n-1)}]^{\mathrm{T}}$，$k_1,\cdots,k_{n-1}$满足 Hurwitz(霍尔伍兹)多项式条件。

将滑模控制律设计为

$$u(t)=\frac{1}{g(\boldsymbol{x},t)}\left[-f(\boldsymbol{x},t)+\sum_{i=1}^{n-1}k_ie^{(i)}-d(t)+x_d^{(n)}-u_{sw}\right] \tag{4.88}$$

其中 $u_{sw}=\eta\operatorname{sgn}(s)$，$\eta>0$。

由式(4.88)和式(4.86)得

$$\dot{s}(\boldsymbol{x},t)=-u_{sw}=-\eta\operatorname{sgn}(s) \tag{4.89}$$

则
$$s(\boldsymbol{x},t)\cdot\dot{s}(\boldsymbol{x},t)=-\eta|s|\leqslant 0$$

当 f、g 和 d 未知时，控制律式(4.88)不适用。采用模糊系统$\hat{f}$、$\hat{g}$ 及 $\hat{h}$ 逼近 f、g 及 $\eta\operatorname{sgn}(s)$。

采用乘积推理机、单值模糊器和中心平均解模糊器，模糊系统的输出分别为$\hat{f}$、$\hat{g}$ 及 $\hat{h}$。则控制律变为

$$u(t)=\frac{1}{\hat{g}(\boldsymbol{x},t)}\left[-\hat{f}(\boldsymbol{x},t)+\sum_{i=1}^{n-1}k_ie^{(i)}+x_d^{(n)}-\hat{h}(s)\right] \tag{4.90}$$

$$\hat{f}(\boldsymbol{x}|\boldsymbol{\theta}_f)=\boldsymbol{\theta}_f^{\mathrm{T}}\boldsymbol{\xi}(x),\hat{g}(\boldsymbol{x}|\boldsymbol{\theta}_g)=\boldsymbol{\theta}_g^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x}),\hat{h}(s|\boldsymbol{\theta}_h)=\boldsymbol{\theta}_h^{\mathrm{T}}\boldsymbol{\phi}(s) \tag{4.91}$$

其中$\hat{f}(\boldsymbol{x}|\boldsymbol{\theta}_f)$、$\hat{g}(\boldsymbol{x}|\boldsymbol{\theta}_g)$、$\hat{h}(s|\boldsymbol{\theta}_h$ 为式(4.70)形式的模糊系统输出，$\boldsymbol{\xi}(x)$和 $\boldsymbol{\phi}(s)$为式(4.72)形式的模糊向量，向量 $\boldsymbol{\theta}_f^{\mathrm{T}}$，$\boldsymbol{\theta}_g^{\mathrm{T}}$ 和$\boldsymbol{\theta}_h^{\mathrm{T}}$ 根据自适应律而变化。

$$\hat{h}(s|\theta_h^*)=\eta_\Delta\operatorname{sgn}(s) \tag{4.92}$$

$$\eta_\Delta=D+\eta,\eta\geqslant 0 \tag{4.93}$$

$$|d(t)|\leqslant D \tag{4.94}$$

设计自适应律为

$$\dot{\boldsymbol{\theta}}_f=r_1s\boldsymbol{\xi}(\boldsymbol{x}) \tag{4.95}$$

$$\dot{\boldsymbol{\theta}}_g=r_2s\boldsymbol{\xi}(\boldsymbol{x})u \tag{4.96}$$

$$\dot{\boldsymbol{\theta}}_h=r_3s\boldsymbol{\phi}(s) \tag{4.97}$$

证明

定义最优参数为

$$\boldsymbol{\theta}_f^* = \arg\min_{\boldsymbol{\theta}_f \in \Omega_f}[\sup_{x\in \mathbf{R}^n} |\hat{f}(\boldsymbol{x}|\boldsymbol{\theta}_f) - f(\boldsymbol{x},t)|] \tag{4.98}$$

$$\boldsymbol{\theta}_g^* = \arg\min_{\boldsymbol{\theta}_g \in \Omega_g}[\sup_{x\in \mathbf{R}^n} |\hat{g}(\boldsymbol{x}|\boldsymbol{\theta}_g) - g(\boldsymbol{x},t)|] \tag{4.99}$$

$$\boldsymbol{\theta}_h^* = \arg\min_{\boldsymbol{\theta}_h \in \Omega_h}[\sup_{s\in \mathbf{R}^n} \hat{h}(s|\boldsymbol{\theta}_h) - u_{sw}|] \tag{4.100}$$

其中 Ω_f,Ω_g 和 Ω_h 分别为 $\boldsymbol{\theta}_f$,$\boldsymbol{\theta}_g$ 和 $\boldsymbol{\theta}_h$ 的集合。

定义最小逼近误差为

$$w = f(\boldsymbol{x},t) - \hat{f}(\boldsymbol{x}|\boldsymbol{\theta}_f^*) + (g(\boldsymbol{x},t) - \hat{g}(\boldsymbol{x}|\boldsymbol{\theta}_g^*))u \tag{4.101}$$

$$|w| \leqslant w_{\max} \tag{4.102}$$

则

$$\begin{aligned}
\dot{s} &= -\sum_{i=1}^{n} k_i e^{(i)} = -\sum_{i=1}^{n-1} k_i e^{(i)} + x^{(n)} - x_d^{(n)} = -\sum_{i=1}^{n-1} k_i e^{(i)} + f(\boldsymbol{x},t) + g(\boldsymbol{x},t)u(t) + d(t) - x_d^{(n)} \\
&= -\sum_{i=1}^{n-1} k_i e^{(i)} + f(\boldsymbol{x},t) + \hat{g}(\boldsymbol{x},t)u(t) + (g(\boldsymbol{x},t) - \hat{g}(\boldsymbol{x},t))u(t) + d(t) - x_d^{(n)} \\
&= f(\boldsymbol{x},t) - \hat{f}(\boldsymbol{x},t) - \hat{h}(s|\boldsymbol{\theta}_h) + (g(\boldsymbol{x},t) - \hat{g}(\boldsymbol{x},t))u(t) + d(t) \\
&= \hat{f}(\boldsymbol{x}|\boldsymbol{\theta}_f^*) - \hat{f}(\boldsymbol{x},t) - \hat{h}(s|\boldsymbol{\theta}_h) + (\hat{g}(\boldsymbol{x}|\boldsymbol{\theta}_g^*) - \hat{g}(\boldsymbol{x},t))u(t) + d(t) + w + \hat{h}(s|\boldsymbol{\theta}_h^*) - \hat{h}(s|\boldsymbol{\theta}_h^*) \\
&= \boldsymbol{\varphi}_f^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x}) + \boldsymbol{\varphi}_g^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})u(t) + \boldsymbol{\varphi}_h^{\mathrm{T}}\boldsymbol{\phi}(s) + d(t) + w - \hat{h}(s|\boldsymbol{\theta}_h^*)
\end{aligned} \tag{4.103}$$

其中 $k_n = 1$,$\boldsymbol{\varphi}_f = \boldsymbol{\theta}_f^* - \boldsymbol{\theta}_f$,$\boldsymbol{\varphi}_g = \boldsymbol{\theta}_g^* - \boldsymbol{\theta}_g$,$\boldsymbol{\varphi}_h = \boldsymbol{\theta}_h^* - \boldsymbol{\theta}_h$。

定义 Lyapunov 函数

$$V = \frac{1}{2}\left(s^2 + \frac{1}{r_1}\boldsymbol{\varphi}_f^{\mathrm{T}}\boldsymbol{\varphi}_f + \frac{1}{r_2}\boldsymbol{\varphi}_g^{\mathrm{T}}\boldsymbol{\varphi}_g + \frac{1}{r_3}\boldsymbol{\varphi}_h^{\mathrm{T}}\boldsymbol{\varphi}_h\right) \tag{4.104}$$

其中 r_1,r_2 及 r_3 为正常数。则

$$\begin{aligned}
\dot{V} &= s\dot{s} + \frac{1}{r_1}\boldsymbol{\varphi}_f^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_f + \frac{1}{r_2}\boldsymbol{\varphi}_g^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_g + \frac{1}{r_3}\boldsymbol{\varphi}_h^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_h \\
&= s[\boldsymbol{\varphi}_f^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x}) + \boldsymbol{\varphi}_g^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})u(t) + \boldsymbol{\varphi}_h^{\mathrm{T}}\boldsymbol{\phi}(s) + d(t) + w - \hat{h}(s|\boldsymbol{\theta}_h^*)] + \frac{1}{r_1}\boldsymbol{\varphi}_f^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_f + \frac{1}{r_2}\boldsymbol{\varphi}_g^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_g + \frac{1}{r_3}\boldsymbol{\varphi}_h^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_h \\
&= s\boldsymbol{\varphi}_f^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x}) + \frac{1}{r_1}\boldsymbol{\varphi}_f^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_f + s\boldsymbol{\varphi}_g^{\mathrm{T}}\boldsymbol{\xi}(\boldsymbol{x})u(t) + \frac{1}{r_2}\boldsymbol{\varphi}_g^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_g + s\boldsymbol{\varphi}_h^{\mathrm{T}}\boldsymbol{\phi}(s) + \frac{1}{r_3}\boldsymbol{\varphi}_h^{\mathrm{T}}\dot{\boldsymbol{\varphi}}_h \\
&\quad + s[d(t) - \hat{h}(s|\boldsymbol{\theta}_h^*)] + sw
\end{aligned} \tag{4.105}$$

由于

$$\hat{h}(s|\boldsymbol{\theta}_h^*) = \eta_\Delta \operatorname{sgn}(s)$$

则

$$\begin{aligned}
\dot{V} &= \frac{1}{r_1}\boldsymbol{\varphi}_f^{\mathrm{T}}[r_1 s\,\boldsymbol{\xi}(\boldsymbol{x}) + \dot{\boldsymbol{\varphi}}_f] + \frac{1}{r_2}\boldsymbol{\varphi}_g^{\mathrm{T}}[r_2 s\,\boldsymbol{\xi}(\boldsymbol{x})u(t) + \dot{\boldsymbol{\varphi}}_g] + \frac{1}{r_3}\boldsymbol{\varphi}_h^{\mathrm{T}}[r_3 s\,\boldsymbol{\phi}(s) + \dot{\boldsymbol{\varphi}}_h] + sd(t) \\
&\quad + sw - (D + \eta)|s| \\
&\leqslant \frac{1}{r_1}\boldsymbol{\varphi}_f^{\mathrm{T}}[r_1 s\,\boldsymbol{\xi}(\boldsymbol{x}) + \dot{\boldsymbol{\varphi}}_f] + \frac{1}{r_2}\boldsymbol{\varphi}_g^{\mathrm{T}}[r_2 s\,\boldsymbol{\xi}(\boldsymbol{x})u(t) + \dot{\boldsymbol{\varphi}}_g] + \frac{1}{r_3}\boldsymbol{\varphi}_h^{\mathrm{T}}[r_3 s\,\boldsymbol{\phi}(s) + \dot{\boldsymbol{\varphi}}_h] + sw - \eta|s|
\end{aligned} \tag{4.106}$$

其中 $\dot{\boldsymbol{\varphi}}_f=-\dot{\boldsymbol{\theta}}_f$，$\dot{\boldsymbol{\varphi}}_g=-\dot{\boldsymbol{\theta}}_g$，$\dot{\boldsymbol{\varphi}}_h=-\dot{\boldsymbol{\theta}}_h$。

将式(4.95)～式(4.97)代入式(4.106)，得

$$\dot{V}\leqslant sw-\eta|s| \tag{4.107}$$

根据模糊逼近理论，自适应模糊系统可实现使逼近误差 w 非常小。因此

$$\dot{V}\leqslant 0 \tag{4.108}$$

4.6.3　仿真实例

被控对象取单级倒立摆，其动态方程如下：

$$\begin{aligned}&\dot{x}_1=x_2\\&\dot{x}_2=\frac{g\sin x_1-mlx_2^2\cos x_1\sin x_1/(m_c+m)}{l[4/3-m\cos^2 x_1/(m_c+m)]}+\frac{\cos x_1/(m_c+m)}{l[4/3-m\cos^2 x_1/(m_c+m)]}u\end{aligned} \tag{4.109}$$

其中 x_1 和 x_2 分别为摆角和摆速；$g=9.8\text{m/s}^2$；m_c 为小车质量，$m_c=1\text{kg}$；m 为摆杆质量，$m=0.1\text{kg}$；l 为摆长的一半，$l=0.5\text{m}$；u 为控制输入。

位置指令为 $x_d(t)=0.1\sin(\pi t)$，切换函数为 $s=-k_1e-\dot{e}$，$k_1=5$。

取5种隶属函数进行模糊化：$\mu_{NM}(x_i)=\exp[-((x_i+\pi/6)/(\pi/24))^2]$，$\mu_{NS}(x_i)=\exp[-((x_i+\pi/12)/(\pi/24))^2]$，$\mu_{ZO}(x_i)=\exp[-(x_i/(\pi/24))^2]$，$\mu_{PS}(x_i)=\exp[-((x_i-\pi/12)/(\pi/24))^2]$，$\mu_{PM}(x_i)=\exp[-((x_i-\pi/6)/(\pi/24))^2]$。则用于逼近 f 和 g 的模糊规则分别有25条。

定义切换函数 $s(t)$ 的隶属函数为 $\mu_{NM}(s)=\dfrac{1}{1+\exp(5(s+3))}$，$\mu_{ZO}(s)=\exp(-s^2)$，$\mu_{PM}(s)=\dfrac{1}{1+\exp(5(s-3))}$。

设 $\boldsymbol{\theta}_f^{\mathrm{T}}$ 和 $\boldsymbol{\theta}_g^{\mathrm{T}}$ 为25×1向量，$\boldsymbol{\theta}_h^{\mathrm{T}}$ 为3×1向量。向量 $\boldsymbol{\theta}_f^{\mathrm{T}}$，$\boldsymbol{\theta}_g^{\mathrm{T}}$ 和 $\boldsymbol{\theta}_h^{\mathrm{T}}$ 的初始值为0.10。采用控制律式(4.90)，倒立摆初始状态为$[\pi/60,0]$。自适应参数取 $r_1=5$、$r_2=1$，$r_3=10$。在程序中，分别用 fsd_1，fsu_1 和 fs_1 表示 $\xi_1(x)$ 的分子、分母及 $\xi_1(x)$；分别用 fsd_2，fsu_2 和 fs_2 表示 $\xi_2(x)$ 的分子、分母及 $\xi_2(x)$。仿真结果如图4-49～图4-52所示。

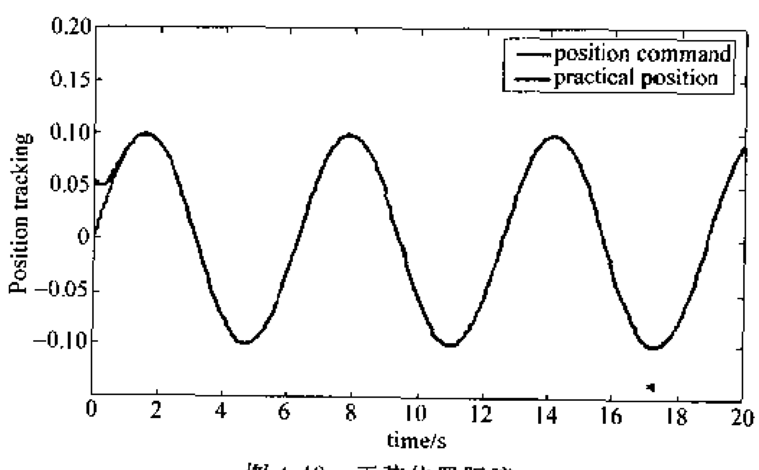


图4-49　正弦位置跟踪

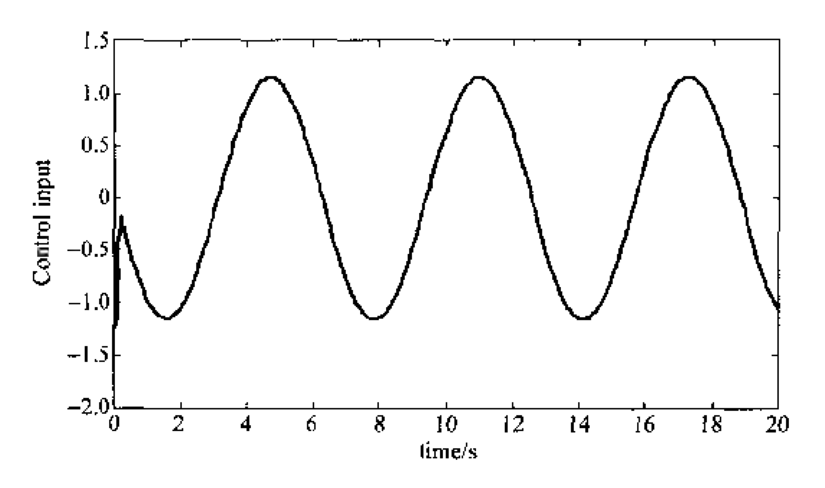


图 4-50　控制输入

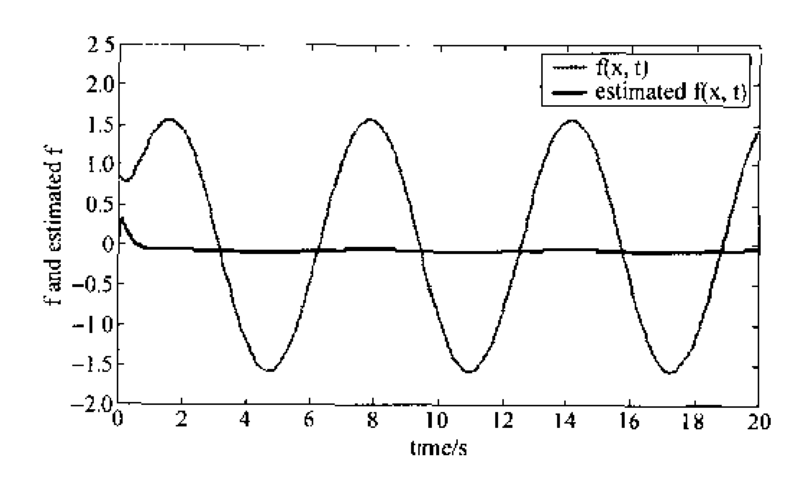


图 4-51　$f(\boldsymbol{x},t)$及$\hat{f}(\boldsymbol{x},t)$的变化

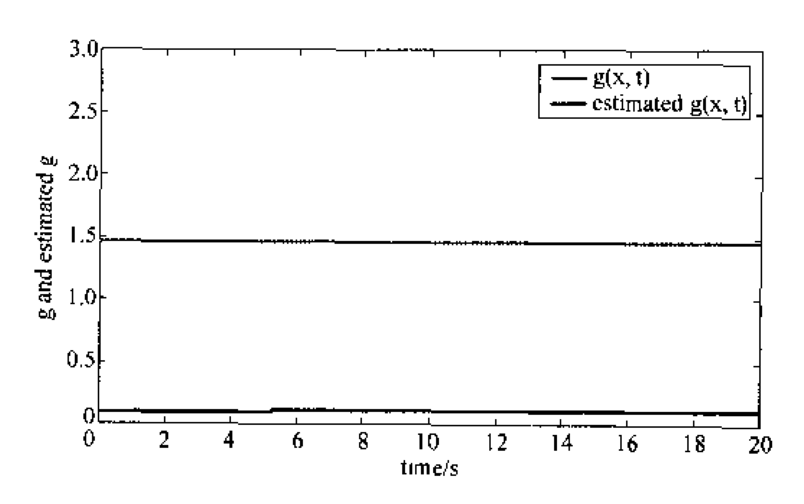


图 4-52　$g(\boldsymbol{x},t)$及$\hat{g}(\boldsymbol{x},t)$的变化

仿真程序如下。

(1) Simulink 主程序(如图 4-53 所示)：chap4_7sim.mdl

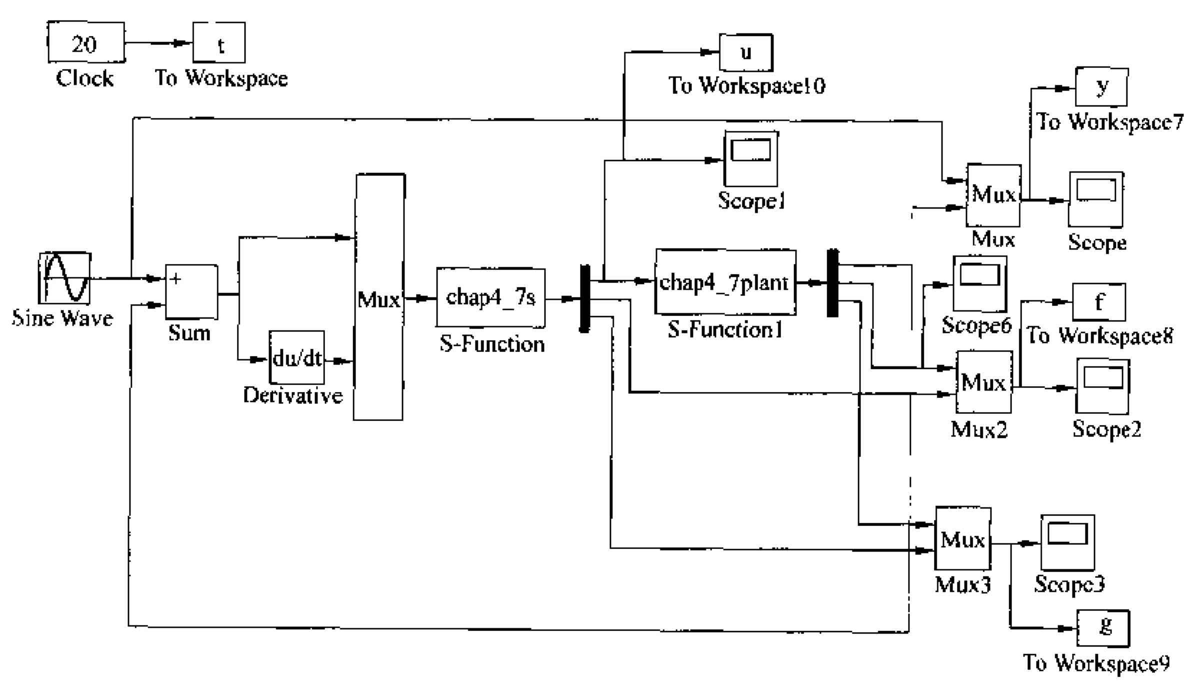


图 4-53 主程序图

(2) 控制器 S 函数:chap4_7s. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
  [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
  sys = mdlDerivatives(t,x,u);
case 3,
  sys = mdlOutputs(t,x,u);
case {2,4,9}
  sys = [];
otherwise
  error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 53;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0.1 * ones(53,1)];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
r1 = 50;r2 = 1;r3 = 10;
```

```
r = 0.1 * sin(t);
dr = 0.1 * cos(t);
ddr = - 0.1 * sin(t);

e = u(1);
de = u(2);
x1 = r - e;
x2 = de - dr;

k1 = 3;
s = - (k1 * e + de);

for i = 1:1:25
  thtaf(i,1) = x(i);
end
for i = 1:1:25
  thtag(i,1) = x(i + 25);
end
for i = 1:1:3
  thtah(i,1) = x(i + 50);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fsd1 = 0;
fsd2 = 0;
for l1 = 1:1:5
  gs1 = - [(x1 + pi/6 - (l1 - 1) * pi/12)/(pi/24)]^2;
  u1(l1) = exp(gs1);
end

for l2 = 1:1:5
  gs2 = - [(x2 + pi/6 - (l2 - 1) * pi/12)/(pi/24)]^2;
  u2(l2) = exp(gs2);
end

for l1 = 1:1:5
  for l2 = 1:1:5
    fsu1(5 * (l1 - 1) + l2) = u1(l1) * u2(l2);
    fsd1 = fsd1 + u1(l1) * u2(l2);
  end
end

fs1 = fsu1/(fsd1 + 0.001);

fx1 = thtaf' * fs1';
gx1 = thtag' * fs1' + 0.001;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gs3 = 5 * (s + 3);
u3(1) = 1/(1 + exp(gs3));

u3(2) = exp( - s^2);
```

```
gs3 = 5 * (s - 3);
u3(3) = 1/(1 + exp(gs3));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fsu2 = u3;
for i = 1:1:3
  fsd2 = fsd2 + u3(i);
end
fs2 = fsu2/(fsd2 + 0.001);
h1 = thtah' * fs2';

u = 1/gx1 * ( - fx1 + ddr - h1 + k1 * de);

S1 = r1 * s * fs1;
S2 = r2 * s * fs1 * u;
S3 = r3 * s * fs2;

for i = 1:1:25
  sys(i) = S1(i);
end
for j = 26:1:50
  sys(j) = S2(j - 25);
end

for j = 51:1:53
  sys(j) = S3(j - 50);
end

function sys = mdlOutputs(t,x,u)
r1 = 50;r2 = 1;r3 = 10;

r = 0.1 * sin(t);
dr = 0.1 * cos(t);
ddr = - 0.1 * sin(t);

e = u(1);
de = u(2);
x1 = r - e;
x2 = de - dr;

k1 = 3;
s = - (k1 * e + de);

for i = 1:1:25
  thtaf(i,1) = x(i);
end
for i = 1:1:25
  thtag(i,1) = x(i + 25);
end
for i = 1:1:3
```

```
  thtah(i,1) = x(i + 50);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fsd1 = 0;
fsd2 = 0;
for l1 = 1:1:5
  gs1 = -[(x1 + pi/6 - (l1 - 1) * pi/12)/(pi/24)]^2;
  u1(l1) = exp(gs1);
end

for l2 = 1:1:5
  gs2 = -[(x2 + pi/6 - (l2 - 1) * pi/12)/(pi/24)]^2;
  u2(l2) = exp(gs2);
end

for l1 = 1:1:5
  for l2 = 1:1:5
    fsu1(5 * (l1 - 1) + l2) = u1(l1) * u2(l2);
    fsd1 = fsd1 + u1(l1) * u2(l2);
  end
end

fs1 = fsu1/(fsd1 + 0.001);

fx1 = thtaf' * fs1';
gx1 = thtag' * fs1' + 0.001;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gs3 = 5 * (s + 3);
u3(1) = 1/(1 + exp(gs3));

u3(2) = exp( - s^2);

gs3 = 5 * (s - 3);
u3(3) = 1/(1 + exp(gs3));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fsu2 = u3;
for i = 1:1:3
  fsd2 = fsd2 + u3(i);
end
fs2 = fsu2/(fsd2 + 0.001);
h1 = thtah' * fs2';

ut = 1/gx1 * ( - fx1 + ddr - h1 + k1 * de);

sys(1) = ut;
sys(2) = fx1;
sys(3) = gx1;
```

(3) 被控对象 S 函数:chap4_7plant.m

```
%S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [pi/60 0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)

g = 9.8;
mc = 1.0;
m = 0.1;
l = 0.5;

S = l * (4/3 - m * (cos(x(1)))^2/(mc + m));
fx = g * sin(x(1)) - m * l * x(2)^2 * cos(x(1)) * sin(x(1))/(mc + m);
fx = fx/S;
gx = cos(x(1))/(mc + m);
gx = gx/S;

sys(1) = x(2);
```

```
sys(2) = fx + gx * u;
function sys = mdlOutputs(t,x,u)

g = 9.8;
mc = 1.0;
m = 0.1;
l = 0.5;

S = l * (4/3 - m * (cos(x(1)))^2/(mc + m));
fx = g * sin(x(1)) - m * l * x(2)^2 * cos(x(1)) * sin(x(1))/(mc + m);
fx = fx/S;
gx = cos(x(1))/(mc + m);
gx = gx/S;

sys(1) = x(1);
sys(2) = fx;
sys(3) = gx;
```

(4) 作图程序：chap4_7plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,f(:,1),'r',t,f(:,2),'b');
xlabel('time(s)');ylabel('f and estiamted f');

figure(4);
plot(t,g(:,1),'r',t,g(:,2),'b');
xlabel('time(s)');ylabel('g and estimated g');
```

4.7 具有积分滑模面的模糊自适应滑模控制

利用滑模控制中的切换函数作为模糊系统的输入，可设计单输入模糊控制器[5]。采用积分滑模面设计切换函数，并采用自适应模糊控制方法，可实现高精度模糊自适应滑模控制[5]。

4.7.1 系统描述

考虑如下 SISO 非线性系统：

$$\ddot{\theta}=f(\theta,t)+g(\theta,t)u(t)+d(t) \tag{4.110}$$

其中 f 和 g 均为未知非线性函数，$g>0$，$d(t)$为外加干扰。

跟踪误差为

$$e(t)=\theta(t)-r(t) \tag{4.111}$$

4.7.2 模糊控制器的设计

定义积分滑模面

$$s(t)=\dot{\theta}(t)-\int_0^t[\ddot{r}(t)-k_1\dot{e}(t)-k_2e(t)]\mathrm{d}t \tag{4.112}$$

其中 k_1 和 k_2 为非零正常数。

如果滑模控制处于理想状态，则 $s(t)=\dot{s}(t)=0$，即

$$\ddot{e}(t)+k_1\dot{e}(t)+k_2e(t)=0 \tag{4.113}$$

通过确定 k_1 和 k_2，跟踪误差 $e(t)$将趋近于零。

相对于传统的二输入模糊控制器而言，可以采用切换函数 $s(t)$作为模糊控制器的输入，构成一个单输入模糊系统，从而大大减少了模糊规则的数量。该模糊控制器的模糊规则形式为

$$\text{Rule } i\text{: IF } s \text{ is } F_s^i \text{ THEN } u \text{ is } \alpha_i \tag{4.114}$$

其中 $i=1,2,\cdots,m$，α_i 和 F_s^i 为模糊集合。

采用重心法进行反模糊化，得到控制器输出：

$$u_{fz}(s)=\frac{\sum_{i=1}^{m}w_i\alpha_i}{\sum_{i=1}^{m}w_i} \tag{4.115}$$

其中 w_i 为第 i 条规则的权值。

4.7.3 仿真实例

采用 S 函数程序 chap4_8s.m 建立模糊系统，并通过命令 persistent 将规则库一直保持在运行过程中。取 flag=1 时，可给出隶属函数图，如图 4-54 和图 4-55 所示。

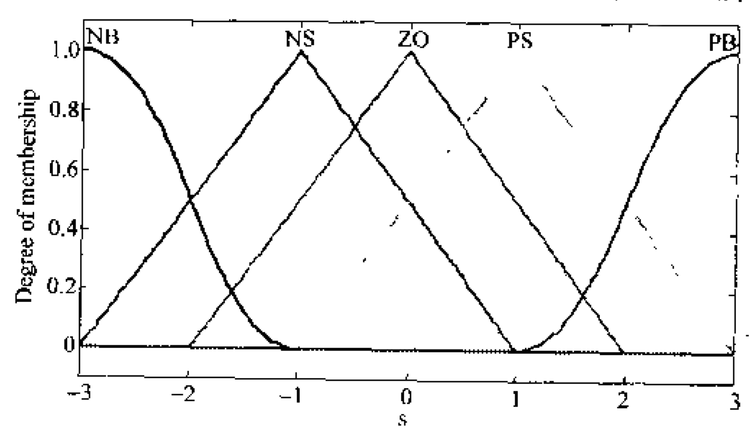


图 4-54 模糊输入隶属函数

在模糊系统中建立如下5条模糊规则：

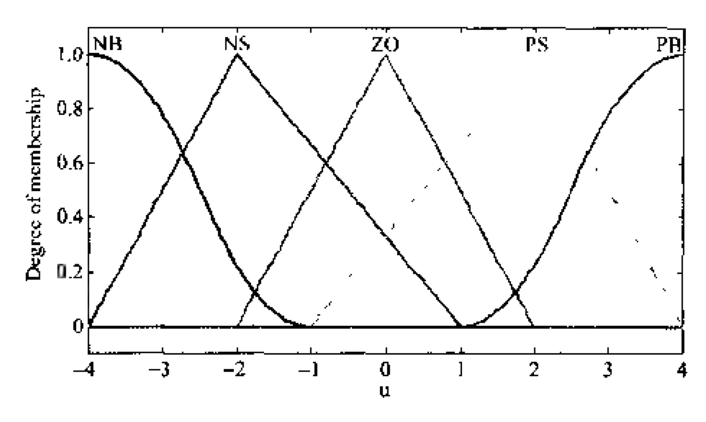


图 4-55　模糊输出隶属函数

(1) If (s is NB) then (u is PB)

(2) If (s is NS) then (u is PS)

(3) If (s is ZO) then (u is ZO)

(4) If (s is PS) then (u is NS)

(5) If (s is PB) then (u is NB)

$s(t)$和$u(t)$的隶属函数采用"负大"(NB)、"负小"(NS)、"零"(ZO)、"正小"(PS)、"正大"(PB)，采用重心法进行反模糊化。

实例 1

被控对象为一线性系统

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=-25x_2+133u(t)+d(t)\end{cases} \tag{4.116}$$

其中$u(t)$为控制输入，$d(t)$为外加干扰。

干扰为$d(t)=200\sin(2\pi t)$，位置指令为$r(t)=0.2\sin\left(\pi t+\frac{\pi}{2}\right)$，取$k_1=150, k_2=200$，则滑模面为$s(t)=\dot{\theta}(t)-\int_0^t[\ddot{r}(t)-150\dot{e}(t)-200e(t)]\mathrm{d}t$。

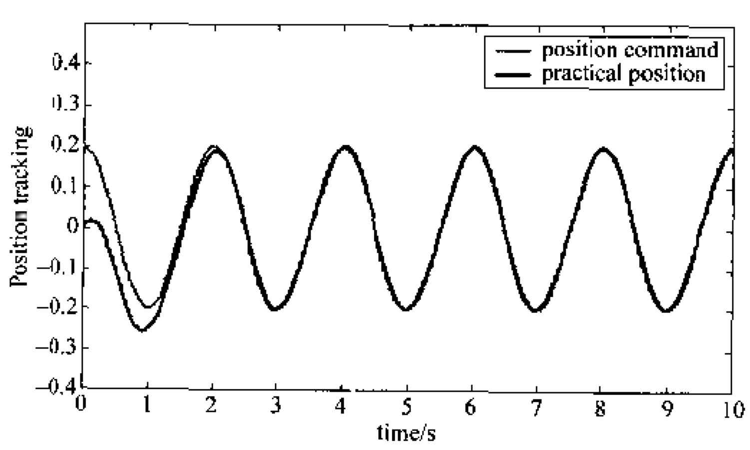


图 4-56　正弦位置跟踪

在程序中，采用 S 函数 chap4_8plant.m 描述线性系统被控对象。取系统的初始状态为 [0,0]，采用模糊滑模控制律式(4.115)，仿真结果如图 4-56 和图 4-57 所示。

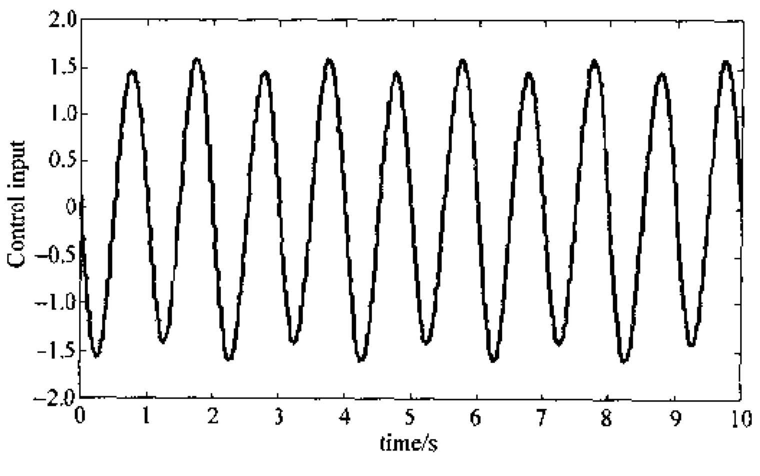


图 4-57　控制输入信号

实例 2

被控对象取单级倒立摆，其动态方程如下：

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=\dfrac{g\sin x_1-mlx_2^2\cos x_1\sin x_1/(m_c+m)}{l(4/3-m\cos^2 x_1/(m_c+m))}+\dfrac{\cos x_1/(m_c+m)}{l(4/3-m\cos^2 x_1/(m_c-m))}u(t)+d(t)\end{cases} \tag{4.117}$$

其中 x_1 和 x_2 分别为摆角和摆速；$g=9.8\text{m/s}^2$；m_c 为小车质量，$m_c=1\text{kg}$；m 为摆杆质量，$m=0.1\text{kg}$；l 为摆长的一半，$l=0.5\text{m}$；u 为控制输入；$d(t)$ 为外加干扰，$d(t)=20\sin(2\pi t)$。

在程序中采用 S 函数 chap4_8plant1.m 描述单级倒立摆被控对象。位置指令为 $r(t)=0.2\sin\left(\pi t+\frac{\pi}{2}\right)$，取 $k_1=10$，$k_2=25$，则滑模面为 $s(t)=\dot{\theta}(t)-\int_0^t[\ddot{r}(t)-10\dot{e}(t)-25e(t)]\mathrm{d}t$。倒立摆初始状态为 $\left[\frac{\pi}{60},0\right]$。采用模糊滑模控制律式(4.115)，仿真结果如图 4-58 所示。

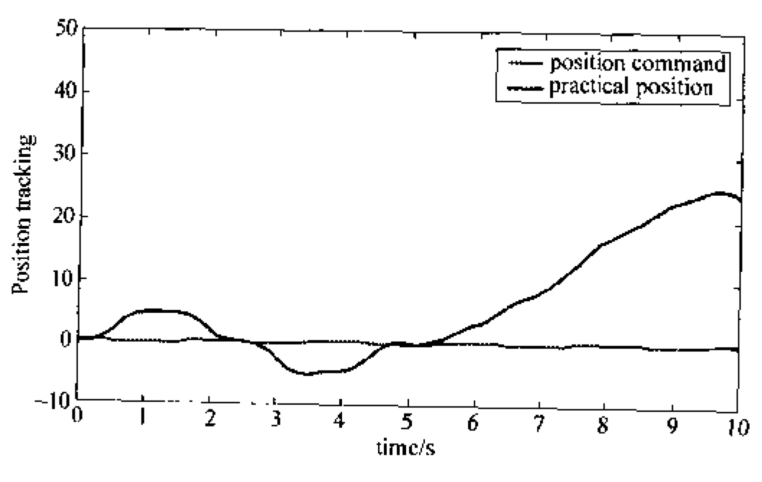


图 4-58　位置跟踪

由仿真结果可见，采用模糊滑模控制律式(4.115)，可实现对线性系统的高精度控制，但很难控制非线性系统。为了解决这一问题，需要采用自适应模糊滑模控制器，见 4.8 节。

仿真程序如下。

(1) Simulink 主程序(如图 4-59 所示)：chap4_8sim.mdl

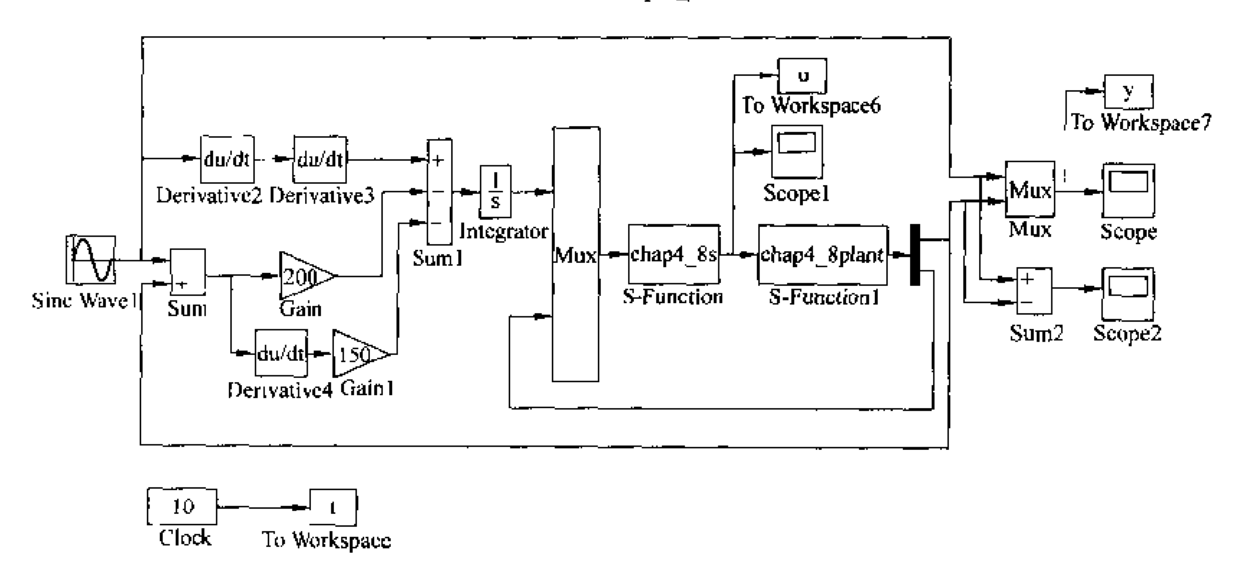


图 4-59　主程序图

(2) 控制器 S 函数：chap4_8s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
   [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
   sys = mdlOutputs(t,x,u);
case {2,4,9}
   sys = [];
otherwise
   error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs  = 1;
sizes.NumInputs  = 2;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 1;
sys  = simsizes(sizes);
x0  = [];
str  = [];
ts  = [0 0];
```

```
function sys = mdlOutputs(t,x,u)
persistent a2

if t = = 0

a = newfis('fuzz_smc');

a = addvar(a,'input','s',[-3,3]);  % Parameter e
a = addmf(a,'input',1,'NB','zmf',[-3,-1]);
a = addmf(a,'input',1,'NS','trimf',[-3,-1,1]);
a = addmf(a,'input',1,'Z','trimf',[-2,0,2]);
a = addmf(a,'input',1,'PS','trimf',[-1,1,3]);
a = addmf(a,'input',1,'PB','smf',[1,3]);

a = addvar(a,'output','u',[-4,4]);  % Parameter u
a = addmf(a,'output',1,'NB','zmf',[-4,-1]);
a = addmf(a,'output',1,'NS','trimf',[-4,-2,1]);
a = addmf(a,'output',1,'Z','trimf',[-2,0,2]);
a = addmf(a,'output',1,'PS','trimf',[-1,2,4]);
a = addmf(a,'output',1,'PB','smf',[1,4]);

rulelist = [1 5 1 1;  % Edit rule base
          2 4 1 1;
          3 3 1 1;
          4 2 1 1;
          5 1 1 1];

a = addrule(a,rulelist);
% showrule(a)              % Show fuzzy rule base

a1 = setfis(a,'DefuzzMethod','centroid');  % Defuzzy
writefis(a1,'fsmc');  % Save fuzzy system as "fsmc.fis"
a2 = readfis('fsmc');

flag = 1;
if flag = = 1
figure(1);
plotmf(a1,'input',1);
figure(2);
plotmf(a1,'output',1);
end

end

s = u(2) - u(1);

ut = evalfis([s],a2);  % Using fuzzy inference

sys(1) = ut;
```

(3) 被控对象 S 函数

实例 1:被控对象 S 函数 chap4_8plant. m

```
%S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag - ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0 0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
dt = 200 * sin(2 * pi * t);

sys(1) = x(2);
sys(2) = -25 * x(2) + 133 * u + dt;

function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

实例 2:被控对象 S 函数 chap4_8plant1. m

```
%S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)
```

```
switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys=mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs  = 2;
sizes.NumInputs  = 1;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 0;

sys=simsizes(sizes);
x0=[pi/60 0];
str=[];
ts=[];

function sys=mdlDerivatives(t,x,u)

g=9.8;
mc=1.0;
m=0.1;
l=0.5;

S=l*(4/3-m*(cos(x(1)))^2/(mc+m));
fx=g*sin(x(1))-m*l*x(2)^2*cos(x(1))*sin(x(1))/(mc+m);
fx=fx/S;
gx=cos(x(1))/(mc+m);
gx=gx/S;

dt=1*20*sin(2*pi*t);

sys(1)=x(2);
sys(2)=fx+gx*u+dt;

function sys=mdlOutputs(t,x,u)
```

```
sys(1) = x(1);
sys(2) = x(2);
```

(4) 作图程序:chap4_8plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');
```

4.8 自适应模糊滑模控制

4.8.1 控制器设计

假设 f,g 及 d 为已知,则根据式(4.113)可得控制律为

$$u^*(t)=g(\boldsymbol{\theta},t)^{-1}[-f(\boldsymbol{\theta},t)-d(t)+\ddot{r}-k_1\dot{e}-k_2e] \tag{4.118}$$

当 f,g 和 $d(t)$ 为未知时,$u^*(t)$ 难于实现,可采用模糊系统逼近 $u^*(t)$。取 α_i 为可调参数,则式(4.115)变为

$$u_{fz}(s,\boldsymbol{\alpha})=\boldsymbol{\alpha}^{T}\boldsymbol{\xi}^{T} \tag{4.119}$$

其中 $\boldsymbol{\alpha}=[\alpha_1\alpha_2\cdots\alpha_m]^T$,$\boldsymbol{\xi}=[\xi_1\xi_2\cdots\xi_m]$。

ξ_i 定义为

$$\xi_i=\frac{w_i}{\sum_{i=1}^{m}w_i} \tag{4.120}$$

根据模糊逼近理论,存在一个最优模糊系统 $u_{fz}(s,\boldsymbol{\alpha}^*)$ 来逼近 $u^*(t)$。

$$u^*(t)=u_{fz}(s,\boldsymbol{\alpha}^*)+\varepsilon=\boldsymbol{\alpha}^{*T}\boldsymbol{\xi}+\varepsilon \tag{4.121}$$

其中 ε 为逼近误差,满足 $|\varepsilon|<E$。

采用模糊系统 u_{fz} 逼近 $u^*(t)$,则

$$u_{fz}(s,\hat{\boldsymbol{\alpha}})=\hat{\boldsymbol{\alpha}}^{T}\boldsymbol{\xi} \tag{4.122}$$

其中 $\hat{\boldsymbol{\alpha}}$ 为 $\boldsymbol{\alpha}^*$ 的估计值。

采用切换控制律 u_{vs} 来补偿 u^* 与 u_{fz} 之间的误差,则总控制律为

$$u(t)=u_{fz}+u_{vs} \tag{4.123}$$

自适应模糊滑模控制系统的结构如图4-60所示。

4.8.2 自适应控制算法设计

根据式(4.121),得

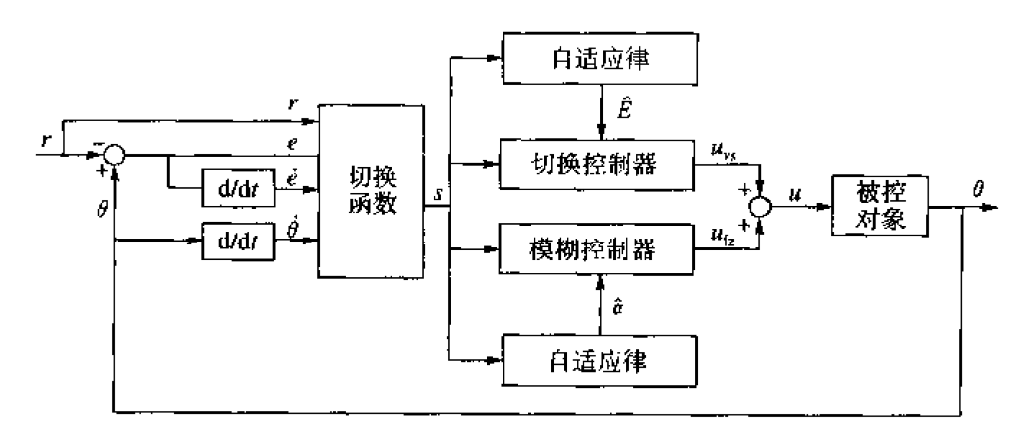


图 4-60 自适应模糊滑模控制系统

$$\tilde{u}_{fz}=\hat{u}_{fz}-u^*=\hat{u}_{fz}-u_{fz}^*-\varepsilon \tag{4.124}$$

定义 $\tilde{\boldsymbol{\alpha}}=\hat{\boldsymbol{\alpha}}-\boldsymbol{\alpha}^*$，则式(4.124)变为

$$\tilde{u}_{fz}=\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\xi}-\varepsilon \tag{4.125}$$

由式(4.112)得

$$\dot{s}(t)=\ddot{e}(t)+k_1\dot{e}(t)+k_2e(t) \tag{4.126}$$

则式(4.118)变为

$$\begin{aligned}u^*(t)&=g(\boldsymbol{\theta},t)^{-1}[-f(\boldsymbol{\theta},t)-d(t)+\ddot{r}+\ddot{e}(t)-\dot{s}(t)]\\&=g(\boldsymbol{\theta},t)^{-1}[-f(\boldsymbol{\theta},t)-d(t)+\ddot{\theta}(t)-\dot{s}(t)]\\&=g(\boldsymbol{\theta},t)^{-1}[g(\boldsymbol{\theta},t)u(t)-\dot{s}(t)]\end{aligned} \tag{4.127}$$

由式(4.123)和式(4.127)得

$$\dot{s}(t)=g(\boldsymbol{\theta},t)[u(t)-u^*(t)]=g(\boldsymbol{\theta},t)[u_{fz}+u_{vs}-u^*(t)] \tag{4.128}$$

定义 Lyapunov 函数

$$V_1(t)=\frac{1}{2}s^2(t)+\frac{g(\boldsymbol{\theta},t)}{2\eta_1}\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}} \tag{4.129}$$

其中 η_1 为正的实数。则

$$\begin{aligned}\dot{V}_1(t)&=s(t)\dot{s}(t)+\frac{g(\boldsymbol{\theta},t)}{\eta_1}\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\dot{\tilde{\boldsymbol{\alpha}}}=s(t)g(\boldsymbol{\theta},t)(u_{fz}+u_{vs}-u^*(t))+\frac{g(\boldsymbol{\theta},t)}{\eta_1}\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\dot{\tilde{\boldsymbol{\alpha}}}\\&=s(t)g(\boldsymbol{\theta},t)(\tilde{u}_{fz}+u_{vs})+\frac{g(\boldsymbol{\theta},t)}{\eta_1}\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\dot{\tilde{\boldsymbol{\alpha}}}=s(t)g(\boldsymbol{\theta},t)(\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\boldsymbol{\xi}-\varepsilon+u_{vs})+\frac{g(\boldsymbol{\theta},t)}{\eta_1}\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\dot{\tilde{\boldsymbol{\alpha}}}\\&=g(\boldsymbol{\theta},t)\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\left(s(t)\boldsymbol{\xi}+\frac{1}{\eta_1}\dot{\tilde{\boldsymbol{\alpha}}}\right)+s(t)g(\boldsymbol{\theta},t)(u_{vs}-\varepsilon)\end{aligned} \tag{4.130}$$

为了达到 $\dot{V}_1\leqslant 0$，采用如下自适应律和切换控制：

$$\dot{\tilde{\boldsymbol{\alpha}}}=\dot{\hat{\boldsymbol{\alpha}}}=-\eta_1 s(t)\boldsymbol{\xi} \tag{4.131}$$

$$u_{vs}=-E(t)\mathrm{sgn}(s(t)) \tag{4.132}$$

则式(4.130)变为

$$\begin{aligned}\dot{V}_1(t)&=-E(t)|s(t)|g(\boldsymbol{\theta},t)-\varepsilon s(t)g(\boldsymbol{\theta},t)\leqslant -E(t)|s(t)|g(\boldsymbol{\theta},t)+|\varepsilon||s(t)|g(\boldsymbol{\theta},t)\\&=-(E(t)-|\varepsilon|)|s(t)|g(\boldsymbol{\theta},t)\leqslant 0\end{aligned} \tag{4.133}$$

在切换控制器中，由于切换增益 $E(t)$ 很难确定，在实际控制中往往通过经验确定。如

果 $E(t)$ 值选得过大，则会产生大的抖振；如果 $E(t)$ 过小，则控制系统不稳定。

用 $\hat{E}(t)$ 代替 $E(t)$，则式(4.132)变为

$$u_{vs}=-\hat{E}(t)\mathrm{sgn}(s(t)) \tag{4.134}$$

其中 $\hat{E}(t)$ 为估计的切换项增益。

定义估计误差为

$$\tilde{E}(t)=\hat{E}(t)-E \tag{4.135}$$

定义 Lyapunov 函数为

$$V(t)=V_1(t)+\frac{g(\boldsymbol{\theta},t)}{2\eta_2}\tilde{E}^2=\frac{1}{2}s^2(t)+\frac{g(\boldsymbol{\theta},t)}{2\eta_1}\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\tilde{\boldsymbol{\alpha}}+\frac{g(\boldsymbol{\theta},t)}{2\eta_2}\tilde{E}^2 \tag{4.136}$$

其中 η_1 和 η_2 为正的常数。则

$$\begin{aligned}\dot{V}(t)&=\dot{V}_1(t)+\frac{g(\boldsymbol{\theta},t)}{\eta_2}\tilde{E}\dot{\tilde{E}}\\&=g(\boldsymbol{\theta},t)\tilde{\boldsymbol{\alpha}}^{\mathrm{T}}\left(s(t)\boldsymbol{\xi}+\frac{1}{\eta_1}\dot{\tilde{\boldsymbol{\alpha}}}\right)+s(t)g(\boldsymbol{\theta},t)(u_{vs}-\varepsilon)+\frac{g(\boldsymbol{\theta},t)}{\eta_2}\tilde{E}\dot{\tilde{E}}\\&=-\hat{E}(t)|s(t)|g(\boldsymbol{\theta},t)-\varepsilon s(t)g(\boldsymbol{\theta},t)+\frac{g(\boldsymbol{\theta},t)}{\eta_2}(\hat{E}(t)-E)\dot{\hat{E}}(t)\end{aligned} \tag{4.137}$$

为了使 $\dot{V}\leqslant 0$，定义自适应律为

$$\dot{\hat{E}}(t)=\eta_2|s(t)| \tag{4.138}$$

则

$$\begin{aligned}\dot{V}(t)&=-\hat{E}(t)|s(t)|g(\boldsymbol{\theta},t)-\varepsilon s(t)g(\boldsymbol{\theta},t)+(\hat{E}(t)-E)|s(t)|g(\boldsymbol{\theta},t)\\&=-\varepsilon s(t)g(\boldsymbol{\theta},t)-E|s(t)|g(\boldsymbol{\theta},t)\leqslant|\varepsilon||s(t)|g(\boldsymbol{\theta},t)-E|s(t)|g(\boldsymbol{\theta},t)\\&=-(E-|\varepsilon|)|s(t)|g(\boldsymbol{\theta},t)\leqslant 0\end{aligned} \tag{4.139}$$

4.8.3 仿真实例

针对滑模面采用以下 5 种隶属函数来模糊化，即 $\mu_{NM}(s)=\exp[-((s+\pi/6)/(\pi/24))^2]$，$\mu_{NS}(s)=\exp[-((s+\pi/12)/(\pi/24))^2]$，$\mu_{ZO}(s)=\exp[-(s/(\pi/24))^2]$，$\mu_{PS}(s)=\exp[-((s-\pi/12)/(\pi/24))^2]$，$\mu_{PM}(s)=\exp[-((s-\pi/6)/(\pi/24))^2]$。采用 5 条规则来逼近 $u^*(t)$。

$\hat{\alpha}$ 和 $\hat{E}$ 的初始值取 0.20，被控对象的初始状态分别取[0,0]和[π/60,0]，控制器参数取 $\eta_1=200$，$\eta_2=0.50$。在程序中，分别用 fsd，fsu 和 fs 表示 $\boldsymbol{\xi}$ 的分子、分母及 $\boldsymbol{\xi}$。

实例 1

以线性系统方程(4.116)为被控对象，在 Simulink 主程序中，被控对象 S 函数程序为 chap4_8plant.m，采用控制律式(4.123)及自适应律式(4.131)和式(4.138)，仿真结果如图 4-61～图 4-63 所示。

实例 2

以倒立摆方程(4.117)为被控对象，在 Simulink 主程序中，被控对象 S 函数程序为 chap4_

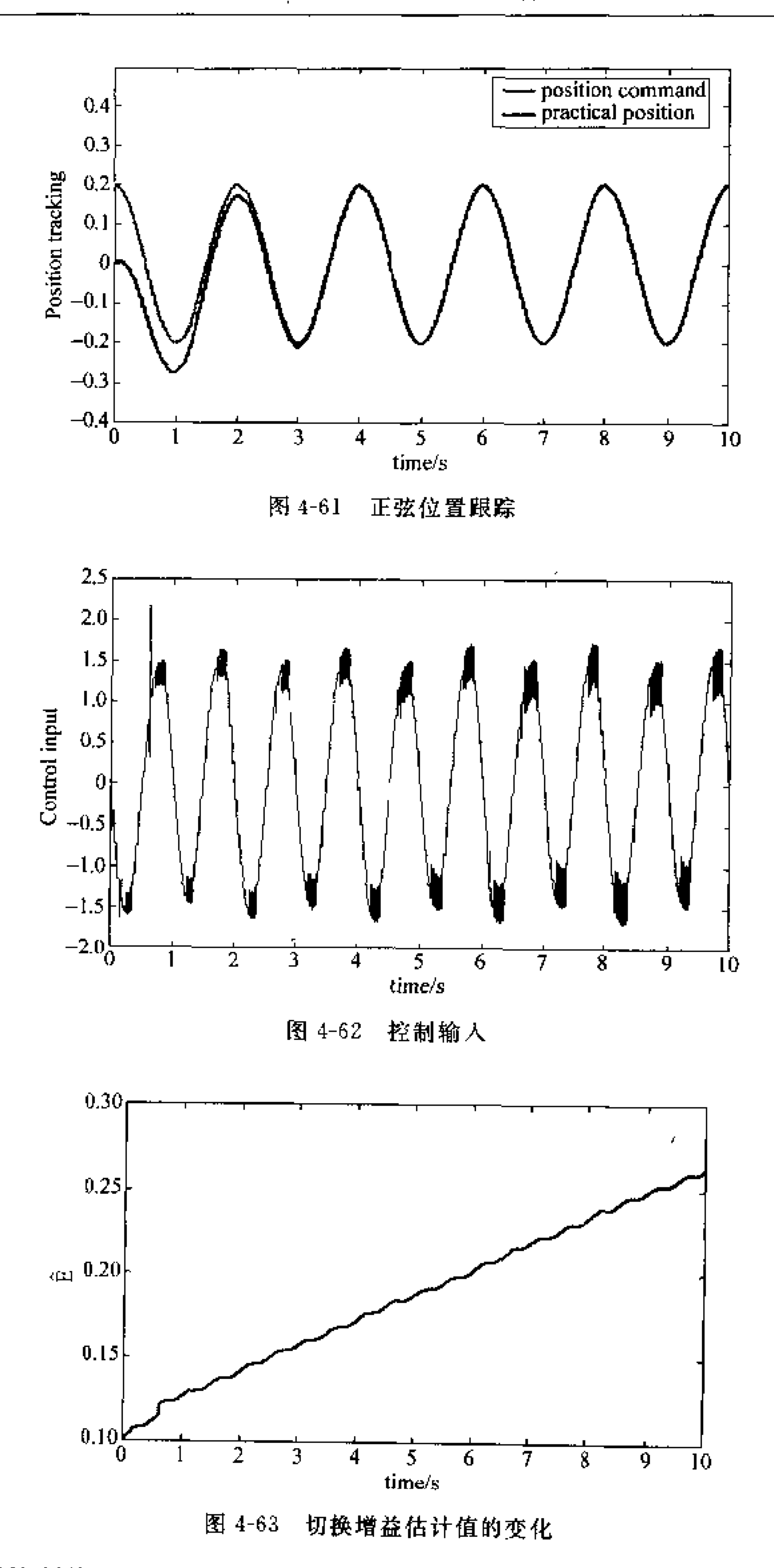


图 4-61 正弦位置跟踪

图 4-62 控制输入

图 4-63 切换增益估计值的变化

8plant1. m，采用控制律式(4.123)及自适应律式(4.131)和式(4.138)，仿真结果如图 4-64～图 4-66 所示。

可见，采用自适应模糊滑模控制可实现对线性系统和非线性系统的高精度控制。

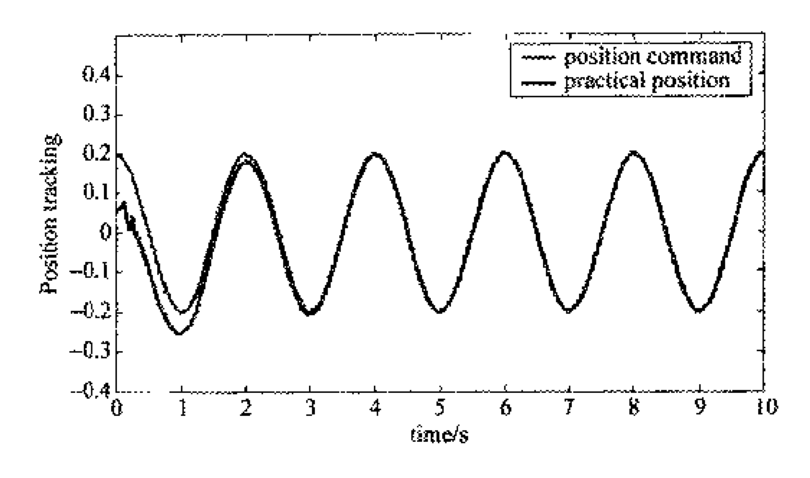


图 4-64 位置跟踪

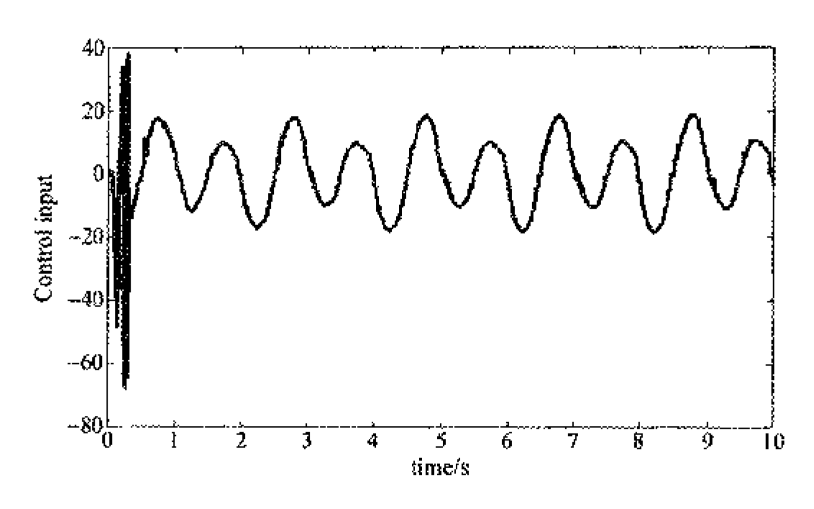


图 4-65 控制输入信号

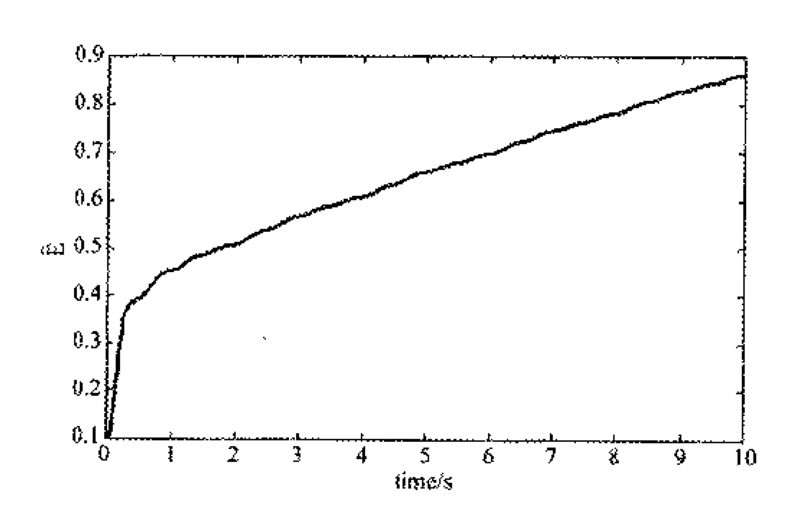


图 4-66 切换项增益估计值的变化

仿真程序如下。

(1) Simulink 主程序(如图 4-67 所示):chap4_9sim.mdl

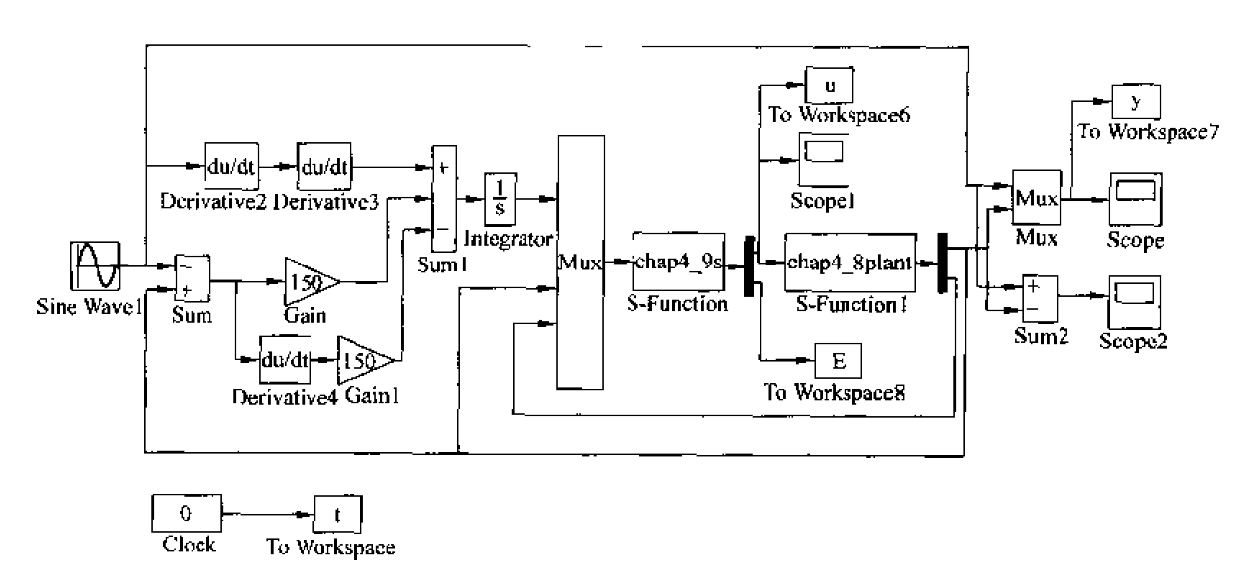


图 4-67 主程序图

(2) 控制器 S 函数程序:chap4_9s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
   [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
   sys = mdlDerivatives(t,x,u);
case 3,
   sys = mdlOutputs(t,x,u);
case {2,4,9}
   sys = [];
otherwise
   error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 6;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0.1 * ones(6,1)];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
r = 0.2 * sin(pi * t + pi/2);
dr = 0.2 * pi * cos(pi * t + pi/2);
```

```
ddr = -0.2*pi*pi*sin(pi*t+pi/2);

y1 = u(2);
y2 = u(3);

e = y1 - r;
de = y2 - dr;

k1 = 10;
k2 = 25;
k = [k2;k1];

s = u(3) - u(1);
for i = 1:1:5
   alfa(i,1) = x(i+1);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fsd = 0;
for l = 1:1:5
   gs = -[(s+pi/6-(l-1)*pi/12)/(pi/24)]^2;
   miu(l) = exp(gs);
end

for l = 1:1:5
   fsu(l) = miu(l);
   fsd = fsd + miu(l);
end

if fsd == 0
   fsd = fsd + 0.01;
end

fs = fsu/fsd;

ufz = alfa'*fs';

E = x(1);
uvs = -E*sign(s);

u = ufz + uvs;

xite1 = 200;
S1 = -xite1*s*fs;

xite2 = 0.50;
sys(1) = xite2*abs(s);
for i = 2:1:6
   sys(i) = S1(i-1);
end
```

```
function sys = mdlOutputs(t,x,u)
r = 0.2 * sin(pi * t + pi/2);
dr = 0.2 * pi * cos(pi * t + pi/2);
ddr = - 0.2 * pi * pi * sin(pi * t + pi/2);

y1 = u(2);
y2 = u(3);

e = y1 - r;
de = y2 - dr;

k1 = 10;
k2 = 25;
k = [k2;k1];

s = u(3) - u(1);
for i = 1:1:5
   alfa(i,1) = x(i + 1);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fsd = 0;
for l = 1:1:5
   gs = -[(s + pi/6 - (l - 1) * pi/12)/(pi/24)]^2;
   miu(l) = exp(gs);
end

for l = 1:1:5
   fsu(l) = miu(l);
   fsd = fsd + miu(l);
end

if fsd == 0
   fsd = fsd + 0.01;
end

fs = fsu/fsd;

ufz = alfa' * fs';

E = x(1);
uvs = - E * sign(s);

ut = ufz + uvs; % FSMC
% ut = - 1500 * e - 150 * de; % PD

sys(1) = ut;
sys(2) = E;
```

(3) 实例 1 被控对象 S 函数程序 chap4_8plant.m

实例 2 被控对象 S 函数程序 chap4_8plant1.m

(4) 作图程序:chap4_9plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,y(:,1)-y(:,2),'r');
xlabel('time(s)');ylabel('Position tracking error');

figure(3);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(4);
plot(t,E(:,1),'r');
xlabel('time(s)');ylabel('E');
```

参考文献

1. 邹云飞,刘金琨,王宗学.模糊滑模控制及在转台伺服系统中的应用.中国控制与决策年会,郑州,2002,369~372
2. Canudas W C, Olsson H, Astrom K J, Lischinsky P. A new model for control of systems with friction. IEEE Transactions on Automatic Control, 1995, 40: 419~425
3. Chen J Y. Expert SMC-based fuzzy control with genetic algorithms. Journal of the Franklin Institute, 1999, 336: 589~610
4. Wang J, Rad A B, Chan P T. Indirect adaptive fuzzy sliding mode control: Part I: fuzzy switching. Fuzzy Sets and systems, 2001, 122:21~30
5. Choi B J, Kwak S W, Kim B K. Design of a single-input fuzzy logic controller and its properties. Fuzzy Sets and Systems, 1999,106: 299~308
6. Wai R J, Lin C M, Hsu C F. Adaptive fuzzy sliding mode control for electrical servo drive. Fuzzy Sets and Systems, 2004, 143: 295~310
7. Wang L X. Fuzzy systems are universal approximators. Proceedings of IEEE Conference on Fuzzy Systems, San diego, 1982, 1163~1170
8. Wang L X. A Course in Fuzzy Systems and Control. Prentice-Hall, Inc, 1997
9. Wang L X. Adaptive Fuzzy Systems and Control: Design and Stability Analysis. Prentice-Hall, Englewood Cliffs, NJ. 1994
10. 刘金琨.智能控制.北京:电子工业出版社.2005

第 5 章　神经滑模控制

5.1　RBF 神经网络逼近

径向基函数(radial basis function,RBF)神经网络是由 J. Moody 和 C. Darken 在 20 世纪 80 年代末提出的一种神经网络,它是具有单隐层的三层前馈网络。由于它模拟了人脑中局部调整、相互覆盖接收域(或称感受野,receptive field)的神经网络结构,因此,RBF 网络是一种局部逼近网络,已证明它能任意精度逼近任意连续函数。

5.1.1　网络结构

RBF 网络是一种三层前向网络,由输入到输出的映射是非线性的,而隐含层空间到输出空间的映射是线性的,从而可以大大加快学习速度并避免局部极小问题。

RBF 网络结构如图 5-1 所示。

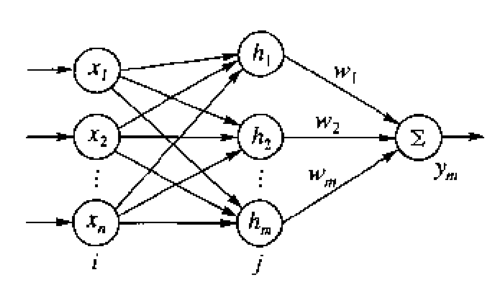


图 5-1　RBF 神经网络结构

5.1.2　逼近算法

RBF 网络的辨识结构如图 5-2 所示。

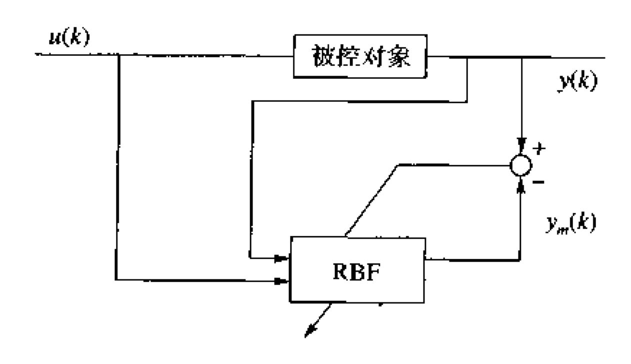


图 5-2　RBF 神经网络逼近

在 RBF 网络结构中，$\boldsymbol{X}=[x_1 x_2 \cdots x_n]^{\mathrm{T}}$ 为网络的输入向量。设 RBF 网络的径向基向量 $\boldsymbol{H}=[h_1 h_2 \cdots h_m]^{\mathrm{T}}$，其中 h_j 为高斯基函数：

$$h_j=\exp\left(-\frac{\|\boldsymbol{X}-\boldsymbol{c}_j\|^2}{2b_j^2}\right) \quad j=1,2,\cdots,m \tag{5.1}$$

其中网络第 j 个节点的中心向量为 $\boldsymbol{c}_j=[c_{j1} c_{j2} \cdots c_{jn}]$。

设网络的基宽向量为

$$\boldsymbol{B}=[b_1 b_2 \cdots b_m]^{\mathrm{T}}$$

其中 b_j 为节点 j 的基宽参数，且为大于零的数。网络的权向量为

$$\boldsymbol{W}=[w_1 w_2 \cdots w_m]^{\mathrm{T}} \tag{5.2}$$

RBF 网络的输出为

$$y_m(t)=w_1 h_1+w_2 h_2+\cdots+w_m h_m \tag{5.3}$$

RBF 网络性能指标函数为

$$E=\frac{1}{2}[y(t)-y_m(t)]^2 \tag{5.4}$$

根据梯度下降法，输出权、节点中心及节点基宽参数的迭代算法如下：

$$w_j(t)=w_j(t-1)+\eta[y(t)-y_m(t)]h_j+\alpha[w_j(t-1)-w_j(t-2)] \tag{5.5}$$

$$\Delta b_j=[y(t)-y_m(t)]w_j h_j \frac{\|\boldsymbol{X}-\boldsymbol{c}_j\|^2}{b_j^3} \tag{5.6}$$

$$b_j(t)=b_j(t-1)+\eta\Delta b_j+\alpha[b_j(t-1)-b_j(t-2)] \tag{5.7}$$

$$\Delta c_{ji}=[y(t)-y_m(t)]w_j \frac{x_j-c_{ji}}{b_j^2} \tag{5.8}$$

$$c_{ji}(t)=c_{ji}(t-1)+\eta\Delta c_{ji}+\alpha[c_{ji}(t-1)-c_{ji}(t-2)] \tag{5.9}$$

其中 η 为学习速率，α 为动量因子。

5.1.3　仿真实例

仿真实例 1：采用 M 语言进行仿真。

使用 RBF 网络逼近下列时变对象：

$$\mathrm{yout}(k)=u(k)^3+\frac{y(k-1)}{1+y(k-1)^2} \quad t\leqslant 1\mathrm{s}$$

$$G_p(s)=\frac{523500}{s^3+87.35s^2+10470s} \quad t>1\mathrm{s}$$

其中对象采样时间为 1ms。

采用 Z 变换对 $G_p(s)$ 进行离散化。经过 Z 变换后的离散化对象为

$$\begin{aligned}\mathrm{yout}(k)=&-\mathrm{den}(2)\mathrm{yout}(k-1)-\mathrm{den}(3)\mathrm{yout}(k-2)-\mathrm{den}(4)\mathrm{yout}(k-3)\\&+\mathrm{num}(2)u(k-1)+\mathrm{num}(3)u(k-2)+\mathrm{num}(4)u(k-3)\end{aligned}$$

在 RBF 网络中，网络输入信号为三个，即 $u(k)$，$\mathrm{yout}(k)$，$\mathrm{yout}(k-1)$，网络初始权值及高斯函数参数初始权值的取值可取随机值，也可通过仿真测试后获得。网络的学习参数取 $\alpha=0.05$，$\eta=0.35$。仿真结果如图 5-3 和图 5-4 所示。

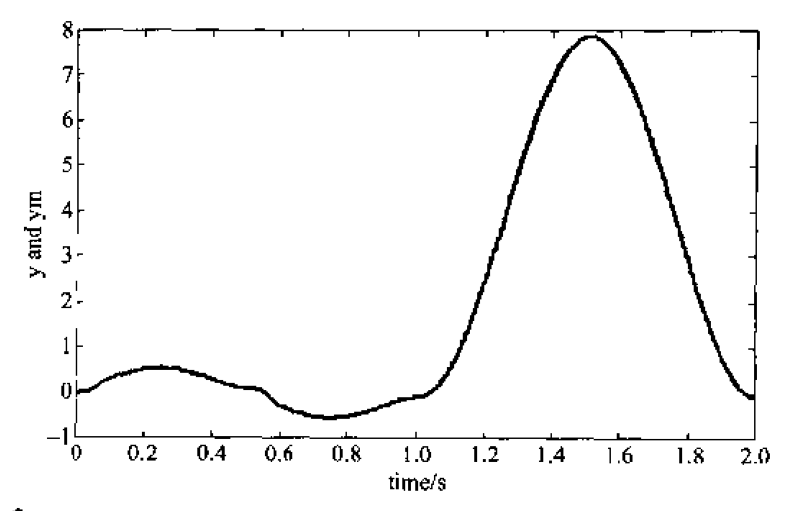


图 5-3　RBF 网络辨识结果

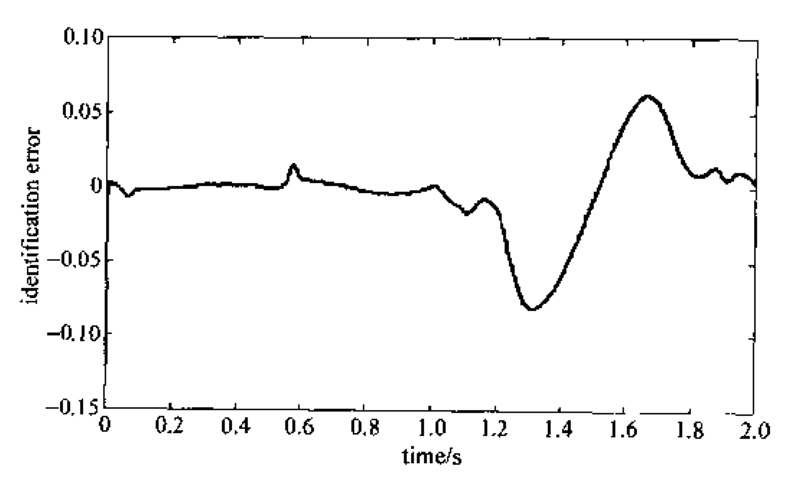


图 5-4　RBF 网络辨识误差

仿真程序：chap5_1. m

```
% RBF identification
clear all;
close all;
sys = tf(5.235e005,[1,87.35,1.047e004,0]);
dsys = c2d(sys,0.001,'z');
[num,den] = tfdata(dsys,'v');

alfa = 0.05;
xite = 0.35;
x = [0,0,0]';

% bi = rands(4,1);
% ci = rands(3,4);
% w = rands(4,1);

bi = [ -2.1268;
   - 0.2870;
     0.8481;
   - 1.7593];
```

```
ci = [ - 0.4568 0.2089 - 0.7983  3.5261;
    - 0.3263  - 0.4569   0.3854  3.0545;
    - 0.3475  - 0.6040  - 0.0453  2.8705];
w = [ 0.3392
      0.0467
    - 0.8302
      4.5379];
w_1 = w;w_2 = w_1;w_3 = w_1;

ci_1 = ci;ci_2 = ci_1;
bi_1 = bi;bi_2 = bi_1;

y_1 = 0;y_2 = 0;y_3 = 0;
u_1 = 0.0;u_2 = 0.0;u_3 = 0.0;

ts = 0.001;
% Main Program
for k = 1:1:2000

time(k) = k * ts;
u(k) = 0.50 * sin(1 * 2 * pi * k * ts);

if k< = 1000
   yout(k) = u(k)^3 + y_1/(1 + y_1^2);
else
   yout(k) = - den(2) * y_1 - den(3) * y_2 - den(4) * y_3 + num(2) * u_1 + num(3) * u_2 + num(4) *
u_3; % Linear control
end
x(3) = y_2;
x(2) = y_1;
x(1) = u_1;

for j = 1:1:4
   h(j) = exp( - norm(x - ci_1(:,j))^2/(2 * bi_1(j) * bi_1(j)));
end
ymout(k) = w_1' * h';

d_w = 0 * w;d_bi = 0 * bi;d_ci = 0 * ci;

for j = 1:1:4
  d_w(j) = xite * (yout(k) - ymout(k)) * h(j);

d_bi(j) = xite * (yout(k) - ymout(k)) * w_1(j) * h(j) * (bi_1(j)^- 3) * norm(x - ci_1(:,j))^2;
  for i = 1:1:3

d_ci(i,j) = xite * (yout(k) - ymout(k)) * w_1(j) * h(j) * (x(i) - ci_1(i,j)) * (bi_1(j)^- 2);
  end
end
  w = w_1 + d_w + alfa * (w_1 - w_2);
  bi = bi_1 + d_bi + alfa * (bi_1 - bi_2);
  ci = ci_1 + d_ci + alfa * (ci_1 - ci_2);
```

```
u_3 = u_2;
u_2 = u_1;
u_1 = u(k);

y_3 = y_2;
y_2 = y_1;
y_1 = yout(k);

w_2 = w_1;
w_1 = w;

ci_2 = ci_1;
ci_1 = ci;

bi_2 = bi_1;
bi_1 = bi;
end
figure(1);
plot(time,ymout,'r',time,yout,'b');
xlabel('time(s)');
ylabel('y and ym');

figure(2);
plot(time,ymout - yout,'r');
xlabel('time(s)');
ylabel('identification error');
```

仿真实例 2:采用 Simulink 语言进行仿真。

使用 RBF 网络逼近下列对象:

$$G_p(s)=\frac{133}{s^2+25s}$$

在 RBF 网络中,网络输入信号为两个,即 $u(t)$和 yout(t),网络初始权值取[0 0 0 0],高斯函数参数的初始权值取 $\boldsymbol{B}=[3\quad 3\quad 3\quad 3]$,$\boldsymbol{c}_i=\begin{bmatrix}0.1 & 0.1 & 0.1 & 0.1\\0.1 & 0.1 & 0.1 & 0.1\end{bmatrix}$,网络的学习参数取 $\alpha=0.05$,$\eta=0.35$。仿真结果如图 5-5 和图 5-6 所示。

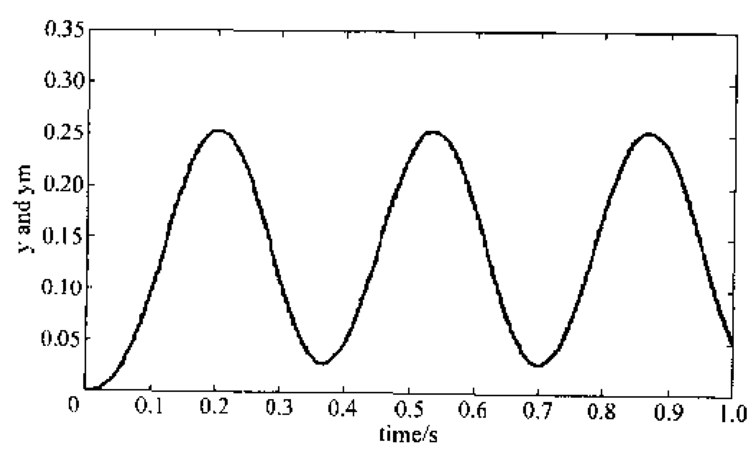


图 5-5 RBF 网络辨识结果

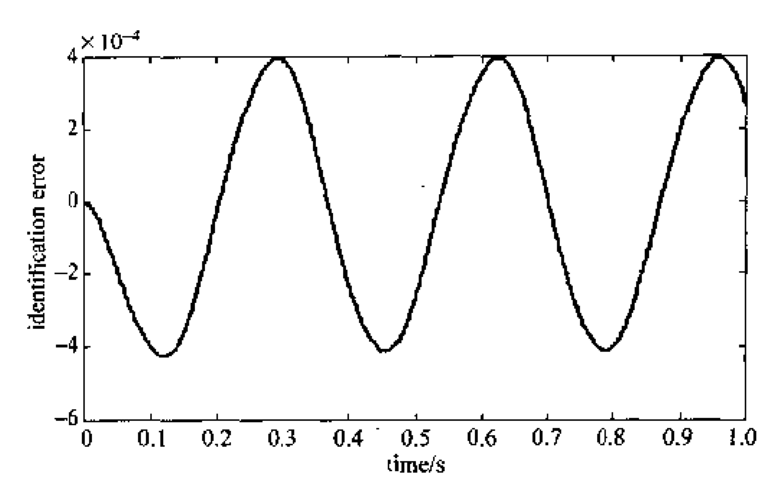


图 5-6　RBF 网络辨识误差

仿真程序：

(1) Simulink 主程序(如图 5-7 所示)：chap5_2sim. mdl

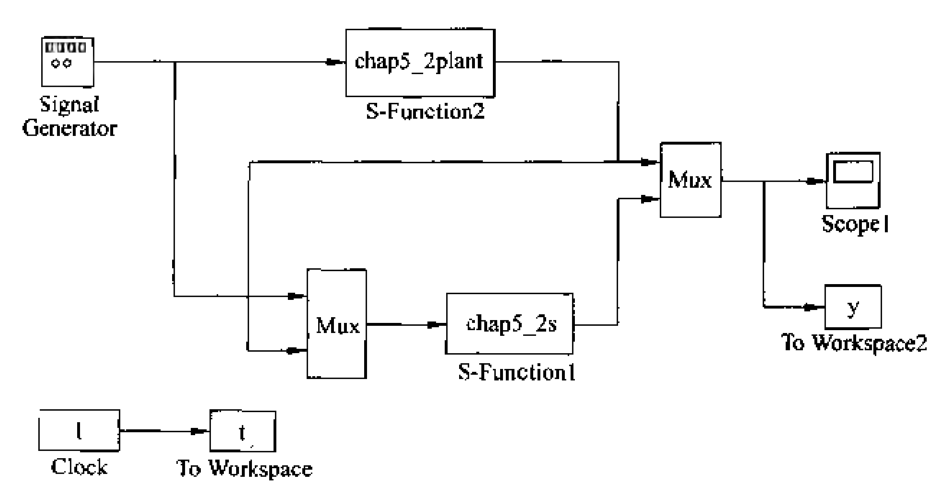


图 5-7　主程序图

(2) 控制器 S 函数：chap5_2s. m

```
% S－function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
```

```
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)
persistent w w_1 w_2 w_3 ci ci_1 ci_2 ci_3 bi bi_1 bi_2 bi_3
alfa = 0.05;
xite = 0.35;

if t == 0
%   bi = rands(4,1);
%   ci = rands(2,4);
    bi = 3 * ones(4,1);
    ci = 0.1 * ones(2,4);
    w = zeros(4,1);
    w_1 = w;w_2 = w_1;w_3 = w_1;
    ci_1 = ci;ci_3 = ci_1;ci_2 = ci_1;
    bi_1 = bi;bi_2 = bi_1;bi_3 = bi_2;
end

ui = u(1);
yout = u(2);

xi = [0,0]';
xi(1) = ui;
xi(2) = yout;

for j = 1:1:4
    h(j) = exp( - norm(xi - ci_1(:,j))^2/(2 * bi_1(j) * bi_1(j)));
end
ymout = w_1' * h';

d_w = 0 * w;d_bi = 0 * bi;d_ci = 0 * ci;

for j = 1:1:4
   d_w(j) = xite * (yout - ymout) * h(j);
   d_bi(j) = xite * (yout - ymout) * w_1(j) * h(j) * (bi_1(j)^ - 3) * norm(xi - ci_1(:,j))^2;
   for i = 1:1:2
```

```
    d_ci(i,j) = xite * (yout - ymout) * w_1(j) * h(j) * (xi(i) - ci_1(i,j)) * (bi_1(j)^-2);
    end
end
    w = w_1 + d_w + alfa * (w_1 - w_2);
    bi = bi_1 + d_bi + alfa * (bi_1 - bi_2);
    ci = ci_1 + d_ci + alfa * (ci_1 - ci_2);

    w_2 = w_1; w_1 = w;
    ci_2 = ci_1; ci_1 = ci;
    bi_2 = bi_1; bi_1 = bi;

sys(1) = ymout;
```

(3) 被控对象 S 函数:chap5_2plant.m

```
%S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
```

```
sys(1) = x(2);
sys(2) = -25 * x(2) + 133 * u;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
```

(4) 作图程序：chap5_2plot. m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time');ylabel('y and ym');

figure(2);
plot(t,y(:,1) - y(:,2),'r');
xlabel('time');ylabel('identification error');
```

5.2 基于 RBF 神经网络的等效滑模控制

5.2.1 系统描述

被控对象为

$$G_p(s)=\frac{1}{Js^2+qs}=\frac{b}{s(s+a)} \tag{5.10}$$

其中 J 为转动惯量，q 为粘性系数。

将式(5.10)转化为状态方程为

$$\dot{\boldsymbol{x}}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}u \tag{5.11}$$

其中 $\boldsymbol{x}=[x_1 \quad x_2]^{\mathrm{T}}$，$\boldsymbol{A}=\begin{bmatrix}0 & 1\\ 0 & -a\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0\\ b\end{bmatrix}$。

将状态方程式(5.11)转化为离散状态方程为

$$\boldsymbol{x}(k+1)=\boldsymbol{A}_1\boldsymbol{x}(k)+\boldsymbol{B}_1u(k) \tag{5.12}$$

其中 $\boldsymbol{x}(k)=[x_1(k) \quad x_2(k)]^{\mathrm{T}}$。

将离散状态方程式(5.12)转化为离散误差状态方程为

$$\boldsymbol{x}_e(k+1)=\boldsymbol{A}_e\boldsymbol{x}_e(k)+\boldsymbol{B}_eu(k)+f(k) \tag{5.13}$$

其中 $f(k)=\begin{bmatrix}-r(k)-A_{12}\dot{r}(k)+r(k+1)\\ -A_{22}\dot{r}(k)+\dot{r}(k+1)\end{bmatrix}=\boldsymbol{R}_1-\boldsymbol{A}_1\boldsymbol{R}$，$\boldsymbol{R}=[r(k) \quad dr(k)]^{\mathrm{T}}$，$\boldsymbol{R}_1=[r(k+1) \quad dr(k+1)]^{\mathrm{T}}$。$\boldsymbol{A}_e=\boldsymbol{A}_1$，$\boldsymbol{B}_e=-\boldsymbol{B}_1$，$\boldsymbol{x}_e(k)=[e(k) \quad de(k)]^{\mathrm{T}}$。$r(k)$ 为位置指令，$dr(k)$ 为位置指令变化率；$e(k)$ 为误差，$de(k)$ 为误差变化率。

5.2.2 等效控制器设计

误差及其变化率为

$$e(k)=r(k)-x_1(k),de(k)=dr(k)-x_2(k) \tag{5.14}$$

切换函数定义为

$$s(k)=ce(k)+de(k)=\boldsymbol{C}\boldsymbol{x}_e(k) \tag{5.15}$$

其中 $\boldsymbol{C}=[c \quad 1]$。

在滑模到达理想状态时，有

$$s(k+1)=s(k) \tag{5.16}$$

由于

$$s(k+1)=\boldsymbol{C}\boldsymbol{x}_e(k+1)=\boldsymbol{C}\boldsymbol{A}_e\boldsymbol{x}_e(k)+\boldsymbol{C}[\boldsymbol{B}_e u(k)+\boldsymbol{f}(k)] \tag{5.17}$$

根据式(5.15)~式(5.17)，得

$$u_{eq}(k)=-(\boldsymbol{C}b)^{-1}[\boldsymbol{C}(\boldsymbol{A}_e-\boldsymbol{I})\boldsymbol{x}_e(k)+\boldsymbol{C}\boldsymbol{f}(k)] \tag{5.18}$$

总的控制律为

$$u(k)=u_{eq}(k)+u_n(k) \tag{5.19}$$

其中 $u_n(k)$ 为 RBF 网络的输出。

5.2.3　神经滑模控制器设计

设 $\boldsymbol{X}=[x_1 x_2 \cdots x_n]^T$ 为网络输入，$\boldsymbol{H}=[h_1 h_2 \cdots h_j \cdots h_m]^T$ 为网络隐含层输出，h_j 为高斯函数。

$$h_j=\exp\left(-\frac{\|\boldsymbol{X}-\boldsymbol{C}_j\|^2}{2b_j^2}\right) \quad j=1,2,\cdots,m \tag{5.20}$$

其中 $\boldsymbol{C}_j=[c_{j1}\cdots c_{jn}]^T$，$\boldsymbol{b}_j=[b_{j1}\cdots b_{jn}]^T$，$m$ 为隐含层个数。

$$\boldsymbol{W}=[w_1 w_2 \cdots w_j \cdots w_m]^T \tag{5.21}$$

网络的输出为

$$u_n(k)=w_1h_1+w_2h_2+\cdots+w_mh_m \tag{5.22}$$

RBF 网络输入为两个，分别为

$$\boldsymbol{x}_n(1)=s(k) \tag{5.23}$$

$$\boldsymbol{x}_n(2)=s(k)-s(k-1) \tag{5.24}$$

选择神经网络学习指标为

$$E(k)=\frac{1}{2}s(k)^2 \tag{5.25}$$

根据式(5.13)和式(5.15)，有

$$\frac{\partial s(k)}{\partial u_n(k)}=\boldsymbol{B}_e(2)$$

根据梯度下降法，神经网络权值学习算法为

$$\Delta w_j=-\frac{\partial E(k)}{\partial w_j}=-s(k)\frac{\partial s(k)}{\partial u_n(k)}\frac{\partial u_n(k)}{\partial w_j}=-s(k)\boldsymbol{B}_e(2)h_j$$

$$w_j(k)=w_j(k-1)+\eta\Delta w_j+\alpha[w_j(k-1)-w_j(k-2)] \tag{5.26}$$

$$\Delta b_j=-\frac{\partial E(k)}{\partial b_j}=-s(k)\frac{\partial s(k)}{\partial u_n(k)}\frac{\partial u_n(k)}{\partial b_j}=-s(k)\boldsymbol{B}_e(2)w_jh_j\frac{\|\boldsymbol{x}_n-\boldsymbol{C}_j\|^2}{b_j^3} \tag{5.27}$$

$$b_j(k)=b_j(k-1)+\eta\Delta b_j+\alpha[b_j(k-1)-b_j(k-2)] \tag{5.28}$$

$$\Delta c_{ji}=-s(k)\frac{\partial s(k)}{\partial u_n(k)}\frac{\partial u_n(k)}{\partial c_{ji}}=-s(k)\boldsymbol{B}_e(2)w_j\frac{x_{nj}-c_{ji}}{b_j^2} \tag{5.29}$$

$$c_{ji}(k)=c_{ji}(k-1)+\eta\Delta c_{ji}+\alpha[c_{ji}(k-1)-c_{ji}(k-2)] \tag{5.30}$$

其中 η 为学习速率，α 为惯性系数。

5.2.4　仿真实例

被控对象为

$$G_p(s)=\frac{1}{Js^2+qs}$$

其中 $J=\frac{1}{133}, q=\frac{25}{133}$。

位置指令为 $r(k)=0.50\sin(6\pi t)$，$t=kt_s$，t_s 为采样时间。设加在控制输入信号上的干扰为 $d(t)=3.0\sin(2\pi t)$。

取 $c=5$，网络结构取 2-6-1，网络学习参数取 $\eta=0.60$，$\alpha=0.05$。网络初始权值及高斯参数的初始值均取随机值。采用控制律式(5.19)。取 $M=1$，控制输入端无干扰，仿真结果如图 5-8～图 5-10 所示。取 $M=2$，控制输入端带干扰 $d(t)$，仿真结果如图 5-11 和图 5-12 所示。

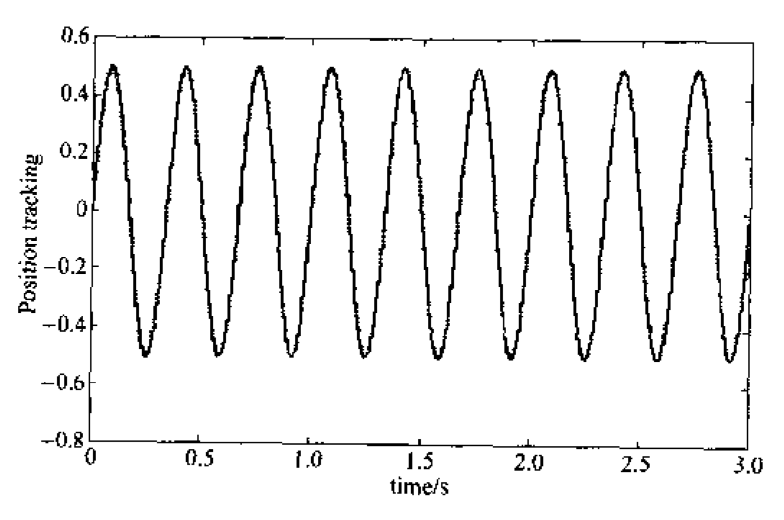


图 5-8　位置跟踪($M=1$)

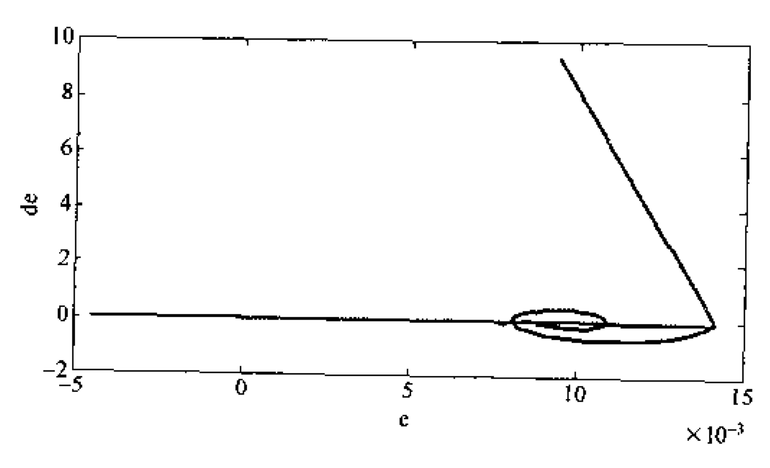


图 5-9　相轨迹($M=1$)

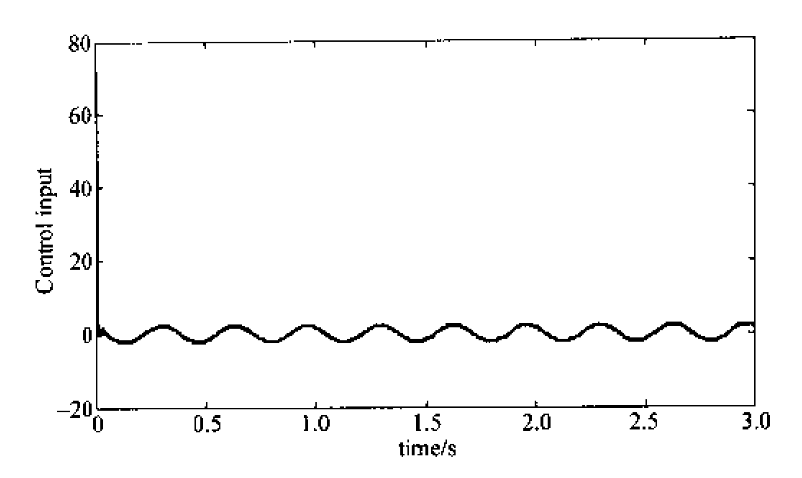


图 5-10　控制输入信号($M=1$)

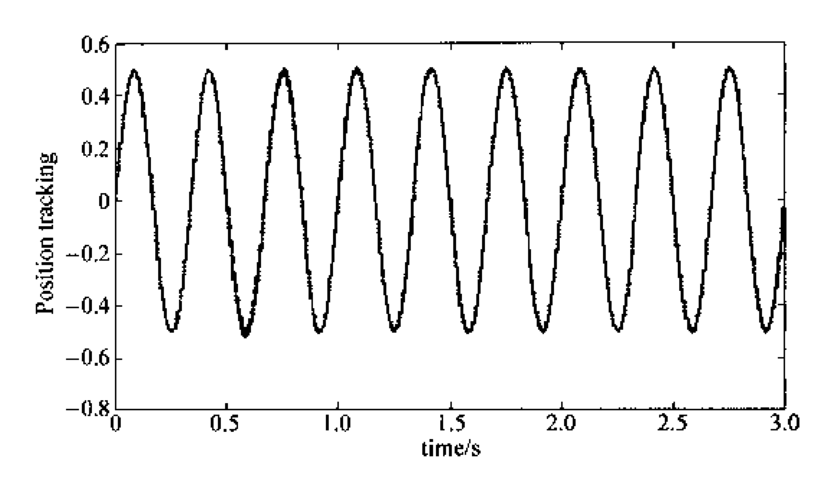


图 5-11　位置跟踪($M=2$)

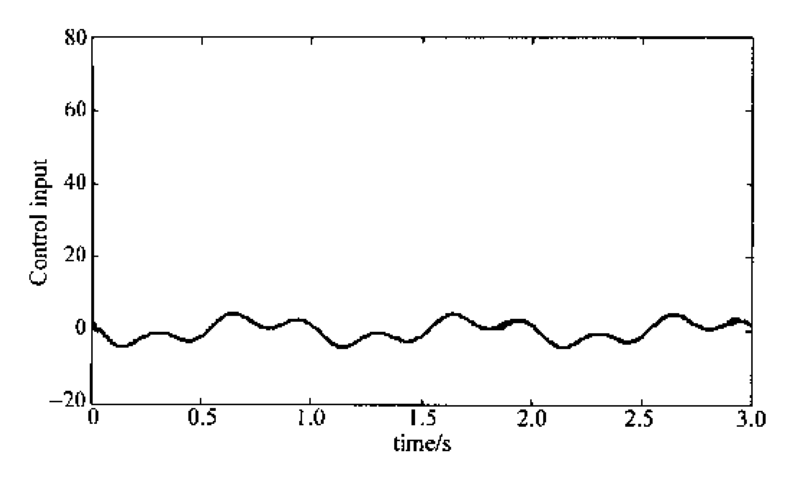


图 5-12　控制输入信号($M=2$)

仿真程序：chap5_3.m

```
%Equivalent Discrete Sliding Mode Control based on RBF neural control
clear all;
close all;
ts=0.001;
```

```
a = 25;
b = 133;
A = [0,1;0,-a];
B = [0;b];
C = [1,0];
D = 0;
[A1,B1,C1,D1] = c2dm(A,B,C,D,ts,'z');
Ae = A1;
Be = -B1;

xite = 0.60;
alfa = 0.05;
x = [0,0]';
xi = [0,0]';

c = rands(2,6);
b = rands(6,1);
w = rands(6,1);

h = [0,0,0,0,0,0]';
c_1 = c;c_2 = c_1;
b_1 = b;b_2 = b_1;
w_1 = w;w_2 = w;

r_1 = 0;r_2 = 0;
s_1 = 0;

for k = 1:1:3000

time(k) = k * ts;

r(k) = 0.50 * sin(6 * pi * k * ts);
cc = 5;
Ce = [cc,1];

% Using extrapolation
  dr(k) = (r(k) - r_1)/ts;
  dr_1 = (r_1 - r_2)/ts;
  r1(k) = 2 * r(k) - r_1;
  dr1(k) = 2 * dr(k) - dr_1;

  R = [r(k);dr(k)];
  R1 = [r1(k);dr1(k)];
  fk = R1 - A1 * R;

  e(k) = r(k) - x(1);
  de(k) = dr(k) - x(2);
  s(k) = cc * e(k) + de(k);
  ds(k) = s(k) - s_1;
```

```
ueq(k) = inv(Ce * Be) * (Ce * (Ae - eye(2)) * [e(k);de(k)] + Ce * fk);

% RBF neural control
xi(1) = s(k);
xi(2) = ds(k);

for j = 1:1:6
   h(j) = exp( - norm(xi - c(:,j))^2/(2 * b(j) * b(j)));
end
un(k) = w' * h;

u(k) = ueq(k) + un(k);

d_w = 0 * w;
for j = 1:1:6
   d_w(j) = - xite * s(k) * Be(2) * h(j);
end

w = w_1 + d_w + alfa * (w_1 - w_2);

d_b = 0 * b;
for j = 1:1:6
   d_b(j) = - xite * s(k) * Be(2) * w(j) * h(j) * (b(j)^-3) * norm(xi - c(:,j))^2;
end
b = b_1 + d_b + alfa * (b_1 - b_2);

d_c = 0 * c;
for j = 1:1:6
   for i = 1:1:2
    d_c(i,j) = - xite * s(k) * Be(2) * w(j) * h(j) * (xi(i) - c(i,j)) * (b(j)^-2);
   end
end
c = c_1 + d_c + alfa * (c_1 - c_2);

dt(k) = 3.0 * sin(2 * pi * k * ts);

M = 2;
if M == 1
   x = A1 * x + B1 * u(k);
elseif M == 2
   x = A1 * x + B1 * (u(k) + dt(k));
end
y(k) = x(1);

% Update Parameters
w_2 = w_1;w_1 = w;
c_2 = c_1;c_1 = c;
b_2 = b_1;b_1 = b;

r_2 = r_1;r_1 = r(k);
```

```
s_1 = s(k);
end
figure(1);
plot(time,r,'b',time,y,'r');
xlabel('Time(second)');ylabel('Position tracking');
figure(2);
plot(time,s);
xlabel('Time(second)');ylabel('Switch function s');
figure(3);
plot(e,de,'b',e,-cc*e,'r');
xlabel('e');ylabel('de');
figure(4);
plot(time,u,'r');
xlabel('Time(second)');ylabel('Control input');
```

5.3 RBF 神经滑模控制

将切换函数作为 RBF 网络的输入，滑模控制器作为 RBF 网络的输出，利用神经网络的学习功能，可实现单入单出的神经滑模控制[1]。

5.3.1 控制器设计

被控对象为

$$\begin{cases}\dot{x}_1 = x_2 \\ \dot{x}_2 = f(\boldsymbol{x}) + bu + d(t)\end{cases} \tag{5.31}$$

设位置指令为 $r(t)$，切换函数设计为

$$s(t) = ce(t) + \dot{e}(t) \tag{5.32}$$

其中

$$e(t) = r(t) - x_1$$

$$\dot{e}(t) = \dot{r}(t) - x_2$$

将滑模控制器设计为 RBF 网络的输出，即

$$u = \sum_{j=1}^{m} w_j \exp\left(-\frac{\| s - \boldsymbol{c}_j \|^2}{b_j}\right) \tag{5.33}$$

其中 m 为隐含层神经元个数。

控制的目标是使 $s(t)\dot{s}(t) \to 0$，则 RBF 网络的权值调整指标为

$$E = s(t)\dot{s}(t) \tag{5.34}$$

则

$$\mathrm{d}w_j = -\eta \frac{\partial E}{\partial w_j(t)} = -\eta \frac{\partial s(t)\dot{s}(t)}{\partial w_j(t)} = -\eta \frac{\partial s(t)\dot{s}(t)}{\partial u(t)} \frac{\partial u(t)}{\partial w_j(t)} \tag{5.35}$$

其中 $\eta > 0$。

由于

$$\frac{\partial s(t)\dot{s}(t)}{\partial u}=s(t)\frac{\partial \dot{s}(t)}{\partial u}=-bs(t) \tag{5.36}$$

$$\frac{\partial u(t)}{\partial w_j(t)}=\exp\left(-\frac{\| s-c_j \|^2}{b_j}\right) \tag{5.37}$$

则 RBF 网络权值学习算法为

$$\mathrm{d}w_j=\gamma s(t)\exp\left(\frac{\| s-c_j \|^2}{b_j}\right)=\gamma s(t)h_j(s) \tag{5.38}$$

RBF 神经滑模控制器结构如图 5-13 所示。

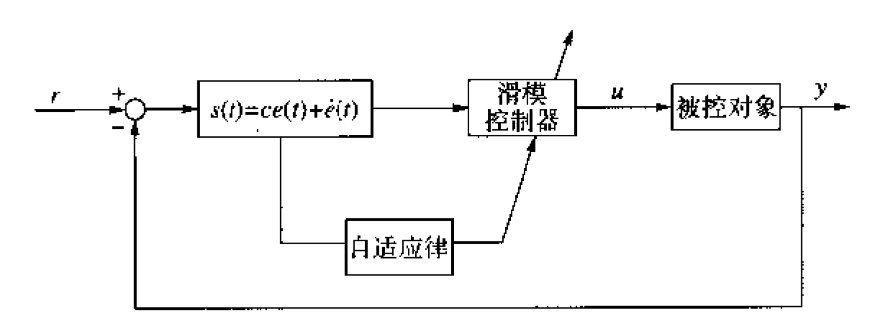


图 5-13　神经滑模控制器结构

5.3.2　仿真实例

被控对象为一线性系统

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=-25x_2+133u(t)+d(t)\end{cases}$$

其中 $u(t)$ 为控制输入，$d(t)$ 为外加干扰。

干扰为 $d(t)=0.10\sin(2\pi t)$，位置指令为 $r(t)=0.5\sin(6\pi t)$。取 $c=25$，则滑模切换函数为 $s(t)=25e(t)+\dot{e}(t)$。神经网络的初始为随机值，网络结构为 1-5-1。高斯函数参数取 $\boldsymbol{c}=[-3\quad -1.5\quad 0\quad 1.5\quad 3]$，$\boldsymbol{b}=[1\quad 1\quad 1\quad 1\quad 1]^{\mathrm{T}}$，采用 persistent 命令实现网络权值的更新。

取系统的初始状态为 $[-0.15\quad 0]$，采用神经滑模控制律式(5.33)，取 $\gamma=1.5$，仿真结果如图 5-14～图 5-16 所示。

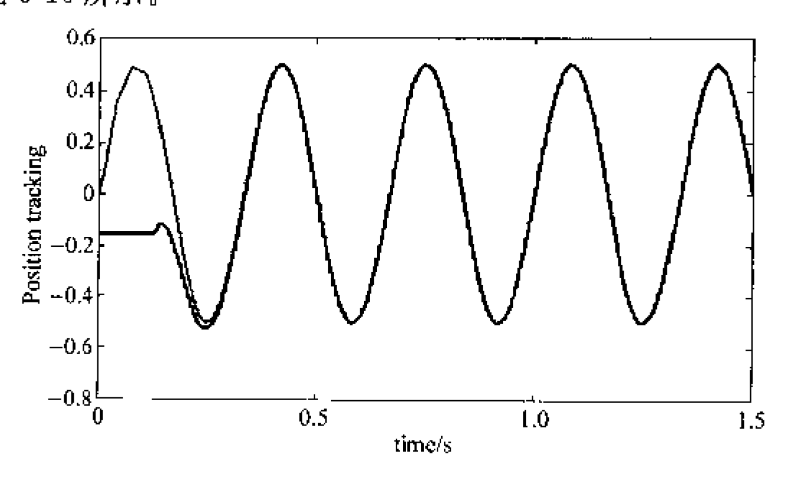


图 5-14　位置跟踪

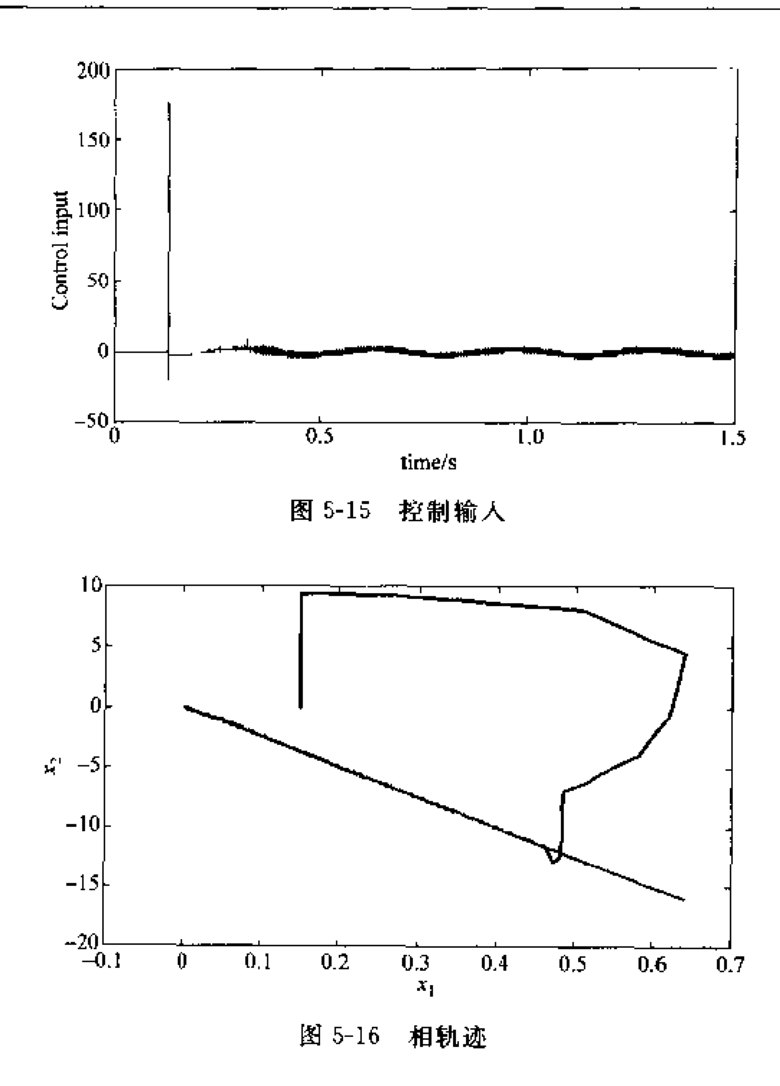


图 5-15 控制输入

图 5-16 相轨迹

仿真程序：

(1) Simulink 主程序(如图 5-17 所示)：chap5_4sim. mdl

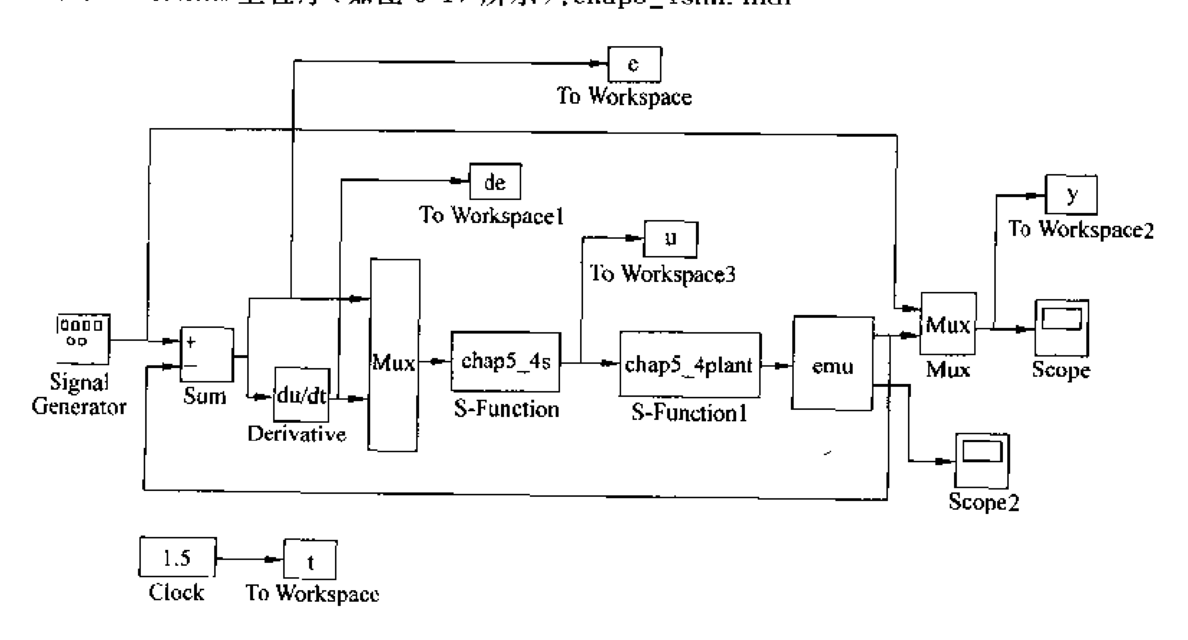


图 5-17 主程序图

(2) 控制器 S 函数:chap5_4s. m

```
%S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 1;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)
persistent w w_1 w_2

gama = 1.5;
alfa = 0.02;
c = [-3 -1.5 0 1.5 3];
b = 1.0 * ones(5,1);
h = [0,0,0,0,0]';

if t == 0
  w = rands(5,1);
  w_1 = w;w_2 = w;
end

cc = 25;
```

```
e = u(1);
de = u(2);
s = cc * e + de;

% RBF neural control
xi = s;

for j = 1:1:5
   h(j) = exp( - norm(xi - c(:,j))^2/(2 * b(j) * b(j)));
end
ut = w' * h;

d_w = 0 * w;
for j = 1:1:5
   d_w(j) = gama * s * h(j);
end

w = w_1 + d_w + alfa * (w_1 - w_2);

% Update Parameters
w_2 = w_1;w_1 = w;

sys(1) = ut;
```

(3) 被控对象 S 函数:chap5_4plant. m

```
% S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 2;
sizes.NumDiscStates   = 0;
```

```
sizes.NumOutputs      = 2;
sizes.NumInputs       = 1;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 0;

sys = simsizes(sizes);
x0 = [-0.15,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
dt = 0.10 * sin(2 * pi * t);

sys(1) = x(2);
sys(2) = -25 * x(2) + 133 * u + dt;
function sys = mdlOutputs(t,x,u)
dt = 0.10 * sin(2 * pi * t);

sys(1) = x(1);
sys(2) = dt;
```

(4) 作图程序:chap5_4plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time');ylabel('position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time');ylabel('control input');

figure(3);
cc = 25;
plot(e,de,'b',e,-cc * e,'r');
xlabel('x1');ylabel('x2');
```

5.4 基于RBF网络上界自适应学习的滑模控制

一般情况下,滑模控制要求系统的各个不确定性的上界值必须已知。而对于实际系统,上界值一般无法测量,采用RBF神经网络可以对干扰的上界进行自适应学习[2],并可降低抖振。

5.4.1 系统描述

被控对象为

$$\ddot{\theta}=f(\theta,\dot{\theta})+bu+f_{\mathrm{d}} \tag{5.39}$$

其中 $b>0$，θ 为角度，$\dot{\theta}$ 为角速度，$f(\cdot)$ 和 b 为未知，f_{d} 是外加干扰信号，控制输入为 u。

将式(5.39)改写为

$$b^{-1}\ddot{\theta}-b^{-1}f(\theta,\dot{\theta})=u(t)+b^{-1}f_{\mathrm{d}} \tag{5.40}$$

令 $M=b^{-1}$，$h(\theta,\dot{\theta})=-b^{-1}f(\theta,\dot{\theta})$，$d(t)=b^{-1}f_{\mathrm{d}}$。则式(5.40)变为

$$M\ddot{\theta}+h(\theta,\dot{\theta})=u(t)+d(t) \tag{5.41}$$

假设 M 和 $h(\theta,\dot{\theta})$ 由确定和不确定两部分组成，即

$$M=M_{\mathrm{n}}+\Delta M \tag{5.42}$$

$$h(\theta,\dot{\theta})=h_{\mathrm{n}}(\theta,\dot{\theta})+\Delta h(\theta,\dot{\theta}) \tag{5.43}$$

其中 M_{n} 和 $h_{\mathrm{n}}(\theta,\dot{\theta})$ 为确定量，ΔM 和 $\Delta h(\theta,\dot{\theta})$ 为不确定量。

由式(5.41)～式(5.43)可得

$$M_{\mathrm{n}}\ddot{\theta}+h_{\mathrm{n}}(\theta,\dot{\theta})=u(t)+\rho(t) \tag{5.44}$$

其中

$$\rho(t)=-\Delta M\ddot{\theta}-\Delta h(\theta,\dot{\theta})+d(t) \tag{5.45}$$

设名义模型为

$$M_{\mathrm{n}}\ddot{\theta}+h_{\mathrm{n}}(\theta,\dot{\theta})=u_1 \tag{5.46}$$

设总控制器为

$$u=u_1+u_0 \tag{5.47}$$

由式(5.44)得

$$M_{\mathrm{n}}\ddot{\theta}+h_{\mathrm{n}}(\theta,\dot{\theta})=u_1+u_0+\rho(t) \tag{5.48}$$

其中 u_1 是名义模型的控制律，u_0 为补偿控制器。

设系统的不确定上界为 $\bar{\rho}(t)$，即

$$|\rho(t)|<\bar{\rho}(t) \tag{5.49}$$

5.4.2　控制器的设计

将系统分成名义模型和不确定系统两部分，采用状态反馈方法对名义模型进行控制，采用 RBF 网络来作为滑模动态补偿器对不确定系统进行控制。该方法无须建立精确模型，可实现对具有外部扰动及参数变化等不定因素的对象进行鲁棒控制。

1. 名义模型控制律设计

设

$$\begin{cases}e_1=\theta-\theta_{\mathrm{d}}\\ e_2=\dot{\theta}-\dot{\theta}_{\mathrm{d}}\end{cases} \tag{5.50}$$

其中，θ_{d} 和 $\dot{\theta}_{\mathrm{d}}$ 分别为 θ 和 $\dot{\theta}$ 的期望值。

当不考虑系统的不确定因素时，由名义模型可得

$$\begin{cases}\dot{e}_1=e_2\\ \dot{e}_2=M_{\mathrm{n}}^{-1}u_1-M_{\mathrm{n}}^{-1}h_{\mathrm{n}}(\theta,\dot{\theta})-\ddot{\theta}_{\mathrm{d}}\end{cases} \tag{5.51}$$

令

$$v=M_n^{-1}[u_1-h_n(\theta,\dot{\theta})]-\ddot{\theta}_d \tag{5.52}$$

$$\begin{bmatrix}\dot{e}_1\\ \dot{e}_2\end{bmatrix}=\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}e_1\\ e_2\end{bmatrix}+\begin{bmatrix}0\\ 1\end{bmatrix}v \tag{5.53}$$

即

$$\dot{\boldsymbol{e}}=\boldsymbol{A}\boldsymbol{e}+\boldsymbol{B}v \tag{5.54}$$

其中，$\boldsymbol{A}=\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0\\ 1\end{bmatrix}$。

设

$$[-k_1 \quad -k_2]\begin{bmatrix}e_1\\ e_2\end{bmatrix}=-k_1e_1-k_2e_2=\boldsymbol{K}\boldsymbol{e}=v \tag{5.55}$$

其中 $\boldsymbol{K}=[-k_1 \quad -k_2]$，$\boldsymbol{e}=[e_1 \quad e_2]^{\mathrm{T}}$。则由式(5.52)的控制律得

$$u_1=M_n(\boldsymbol{K}\boldsymbol{e}+\ddot{\theta}_d)+h_n(\theta,\dot{\theta}) \tag{5.56}$$

将式(5.55)代入式(5.53)，得

$$\begin{bmatrix}\dot{e}_1\\ \dot{e}_2\end{bmatrix}=\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}e_1\\ e_2\end{bmatrix}+\begin{bmatrix}0\\ 1\end{bmatrix}[-k_1 \quad -k_2]\begin{bmatrix}e_1\\ e_2\end{bmatrix} \tag{5.57}$$

故

$$\dot{\boldsymbol{e}}=\boldsymbol{A}_1\boldsymbol{e} \tag{5.58}$$

其中 $\boldsymbol{A}_1=\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K}$。

通过设计状态反馈增益 $\boldsymbol{K}$，利用式(5.58)可以将系统的闭环极点配置到状态空间中的任意一点上。

2. 上界已知时滑模补偿器的设计

当系统存在不确定项时，由式(5.48)得

$$\begin{cases}\dot{e}_1=e_2\\ \dot{e}_2=M_n^{-1}[u_1-h_n(\theta,\dot{\theta})]-\ddot{\theta}_d+M_n^{-1}[u_0+\rho(t)]\end{cases} \tag{5.59}$$

将式(5.52)和式(5.55)代入式(5.59)，得

$$\begin{cases}\dot{e}_1=e_2\\ \dot{e}_2=\boldsymbol{K}\boldsymbol{e}+M_n^{-1}[u_0+\rho(t)]\end{cases} \tag{5.60}$$

滑模面变量 s 定义如下：

$$s=ce_1+e_2 \tag{5.61}$$

则

$$\dot{s}=c\dot{e}_1+\dot{e}_2=ce_2+\dot{e}_2 \tag{5.62}$$

定义 Lyapunov 函数为

$$V=\frac{1}{2}s^2 \tag{5.63}$$

设计补偿控制器 u_0 为

$$u_0=\begin{cases}\dfrac{1}{sM_n^{-1}}w & |s|\neq 0\\ 0 & |s|=0\end{cases} \tag{5.64}$$

其中

$$w=-sce_2-s\boldsymbol{Ke}-|sM_n^{-1}|\bar{\rho}(t) \tag{5.65}$$

稳定性分析：

$$\begin{aligned}\dot{V}&=s\dot{s}=s[ce_2+\boldsymbol{Ke}+M_n^{-1}(u_0+\rho(t))]\\&=s(ce_2+\boldsymbol{Ke})+(sM_n^{-1})u_0+sM_n^{-1}\rho(t)=s(ce_2+\boldsymbol{Ke})+w+sM_n^{-1}\rho(t)\\&=-|sM_n^{-1}|\rho(t)+sM_n^{-1}\rho(t)\leqslant-|sM_n^{-1}|\bar{\rho}(t)+|sM_n^{-1}||\rho(t)|\\&=|sM_n^{-1}|(|\rho(t)|-\bar{\rho}(t))<0\end{aligned} \tag{5.66}$$

式(5.64)也可写为

$$u_0=\begin{cases}-|M_n|(ce_2+\boldsymbol{Ke})-\mathrm{sgn}(s)\mathrm{sgn}(M_n)\bar{\rho}(t) & |s|\neq 0\\ 0 & |s|=0\end{cases} \tag{5.67}$$

3. 基于 RBF 网络的上界自适应学习

一般不确定因素的上界值很难或根本无法预知。根据神经网络的特点，可采用 RBF 神经网络来学习不确定因素的上界值。

RBF 网络的输入为 $\boldsymbol{x}=[\theta \quad \dot{\theta}]$，输出为不确定参数上界的估计值 $\hat{\bar{\rho}}(\boldsymbol{x},\boldsymbol{\omega})$。

$$\hat{\bar{\rho}}(\boldsymbol{x},\boldsymbol{\omega})=\hat{\boldsymbol{\omega}}^{\mathrm{T}}\phi(\boldsymbol{x}) \tag{5.68}$$

其中 $\hat{\boldsymbol{\omega}}^{\mathrm{T}}$ 为 RBF 神经网络的权值，$\phi(x)$为高斯函数。

$$\phi_i(\boldsymbol{x})=\exp\left(-\frac{\|\boldsymbol{x}-m_i\|^2}{\sigma_i^2}\right) \quad i=1,2,3 \tag{5.69}$$

其中 m_i 是第 i 个神经元的中心位置，σ_i 为第 i 个神经元的宽度。

控制律式(5.67)变为

$$u_0=\begin{cases}-|M_n|(ce_2+\boldsymbol{Ke})-\mathrm{sgn}(s)\mathrm{sgn}(M_n)\hat{\bar{\rho}}(t) & |s|\neq 0\\ 0 & |s|=0\end{cases} \tag{5.70}$$

假设 1 设 RBF 网络的最优权值 ω^* 满足：

$$\omega^{*\mathrm{T}}\phi(\boldsymbol{x})-\rho(t)=\varepsilon(\boldsymbol{x})<\varepsilon_1$$

假设 2 不确定参数的上界满足：

$$\bar{\rho}(t)-|\rho(t)|>\varepsilon_0>\varepsilon_1$$

定义 Lyapunov 函数为

$$V=\frac{1}{2}s^2+\frac{1}{2}\eta^{-1}\tilde{\omega}^{\mathrm{T}}\tilde{\omega}$$

其中 $\tilde{\omega}=\omega^*-\hat{\omega}$。

稳定性分析：

采用自适应算法在线调整权值，取

$$\dot{\hat{\boldsymbol{\omega}}}=\eta|sM_n^{-1}|\phi(\boldsymbol{x}) \tag{5.71}$$

并取

$$\eta = |M_n^{-1}|(\varepsilon_0 - \varepsilon_1) > 0 \tag{5.72}$$

$$\begin{aligned}
\dot{V} &= s\dot{s} - \eta^{-1}\tilde{\boldsymbol{\omega}}^{\mathrm{T}}\dot{\hat{\boldsymbol{\omega}}} \\
&= s(ce_2 + \boldsymbol{K}\boldsymbol{e}) + w + sM_n^{-1}\rho(t) - \eta^{-1}\tilde{\boldsymbol{\omega}}^{\mathrm{T}}\dot{\hat{\boldsymbol{\omega}}} \\
&= -|sM_n^{-1}|\hat{\boldsymbol{\omega}}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + sM_n^{-1}\rho(t) - \eta^{-1}\tilde{\boldsymbol{\omega}}^{\mathrm{T}}\dot{\hat{\boldsymbol{\omega}}} \\
&= -|sM_n^{-1}|(\hat{\boldsymbol{\omega}}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + \bar{\rho}(t) - \bar{\rho}(t)) + sM_n^{-1}\rho(t) - \eta^{-1}\tilde{\boldsymbol{\omega}}^{\mathrm{T}}\dot{\hat{\boldsymbol{\omega}}} \\
&\leqslant -|sM_n^{-1}|(\hat{\boldsymbol{\omega}}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) - \rho(t)) - |sM_n^{-1}|(\bar{\rho}(t) - |\rho(t)|) - \eta^{-1}\tilde{\boldsymbol{\omega}}^{\mathrm{T}}\dot{\hat{\boldsymbol{\omega}}} \\
&= -|sM_n^{-1}|(\hat{\boldsymbol{\omega}}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\omega}^{*\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + \varepsilon(\boldsymbol{x})) - |sM_n^{-1}|(\bar{\rho}(t) - |\rho(t)|) \\
&\quad - (\hat{\boldsymbol{\omega}}^{\mathrm{T}} - \boldsymbol{\omega}^{*\mathrm{T}})|sM_n^{-1}|\boldsymbol{\phi}(\boldsymbol{x}) \\
&= -|sM_n^{-1}|\varepsilon(\boldsymbol{x}) - |sM_n^{-1}|(\bar{\rho}(t) - |\rho(t)|) \\
&\leqslant |sM_n^{-1}|\,|\varepsilon(\boldsymbol{x})| - |sM_n^{-1}|(\bar{\rho}(t) - |\rho(t)|) \\
&= |sM_n^{-1}|(|\varepsilon(\boldsymbol{x})| - (\bar{\rho}(t) - |\rho(t)|))
\end{aligned}$$

由假设 1 得

$$|\varepsilon(\boldsymbol{x})| < \varepsilon_1$$

由假设 2 得

$$-(\bar{\rho}(t) - |\rho(t)|) < -\varepsilon_0$$

由上两式得

$$|\varepsilon(\boldsymbol{x})| - (\bar{\rho}(t) - |\rho(t)|) < \varepsilon_1 - \varepsilon_0$$

即

$$\dot{V} \leqslant -|sM_n^{-1}|(\varepsilon_0 - \varepsilon_1) = -\eta|s| \leqslant 0$$

5.4.3　仿真实例

设被控对象为

$$\begin{cases}\dot{x}_1 = x_2 \\ \dot{x}_2 = -25x_2 + 133u + d(t)\end{cases}$$

RBF 网络的输入为$[x_1 \quad x_2]$，权值 $\boldsymbol{w}$ 的初值取$[0.1 \quad 0.1 \quad 0.1]^{\mathrm{T}}$，$\boldsymbol{m}$ 的初值取$-1\sim1$之间的随机数，$\boldsymbol{\sigma}$（程序中用 b 表示）的初值取$[0.2 \quad 0.2 \quad 0.2]^{\mathrm{T}}$。系统初始状态为$[0.5 \quad 0]$，干扰及不确定项 $d(t)$ 取 $0.5\sin(2\pi t)$。

状态反馈控制律中，取 $k_1=100$，$k_2=200$。滑模控制律中，取 $c=30$，位置指令为 $\sin(\pi t)$。当 $M=1$ 时，为上界已知的滑模控制，采用控制律式(5.56)和式(5.64)。当 $M=2$ 时，为基于 RBF 网络上界学习的滑模控制，采用控制律式(5.56)、式(5.70)及自适应律式(5.71)，取 $\varepsilon_0=0.002$，$\varepsilon_1=0.001$。仿真结果如图 5-18～图 5-22 所示。

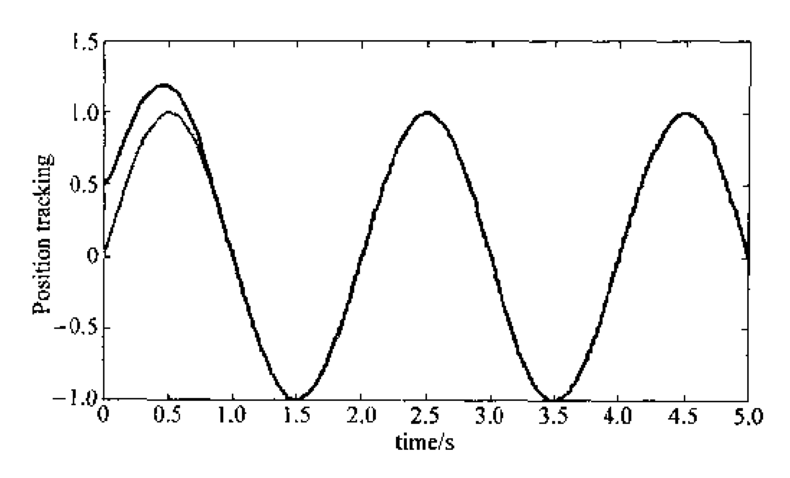


图 5-18 位置跟踪（$M=1$）

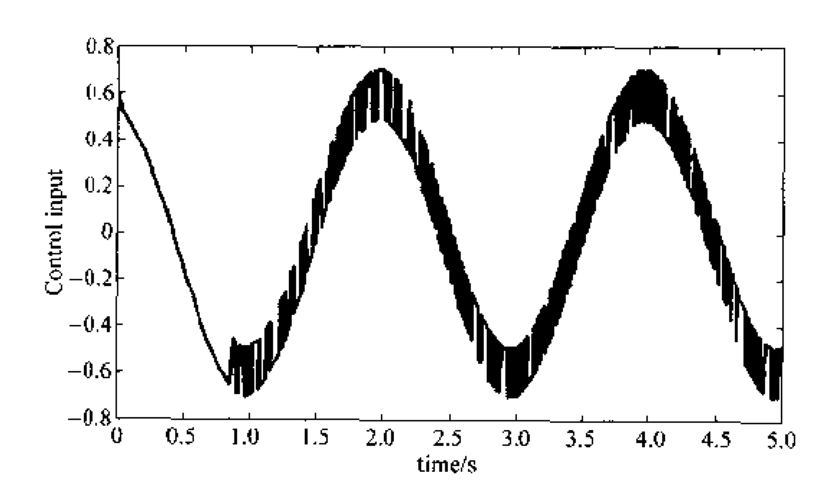


图 5-19 控制输入（$M=1$）

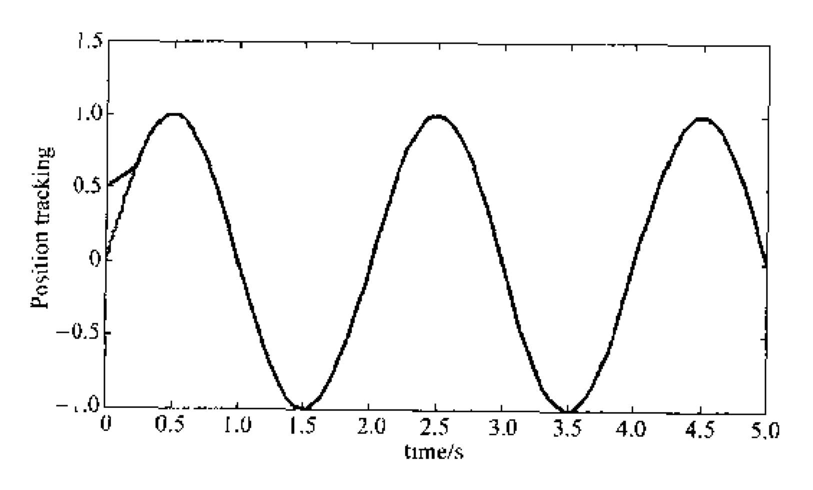


图 5-20 位置跟踪（$M=2$）

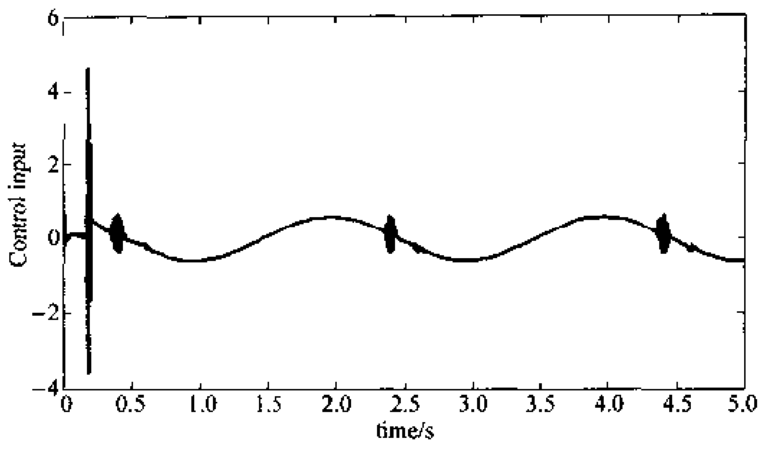


图 5-21　控制输入（$M=2$）

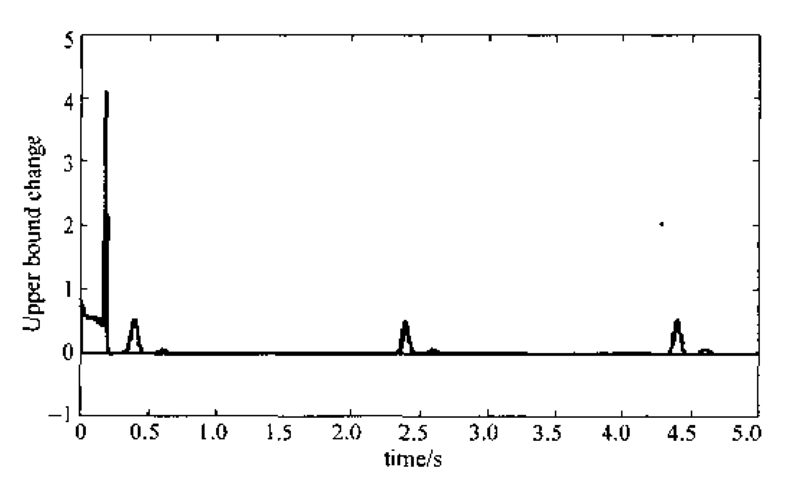


图 5-22　上界的变化（$M=2$）

仿真程序：

（1）Simulink 主程序（如图 5-23 所示）：chap5_5sim. mdl

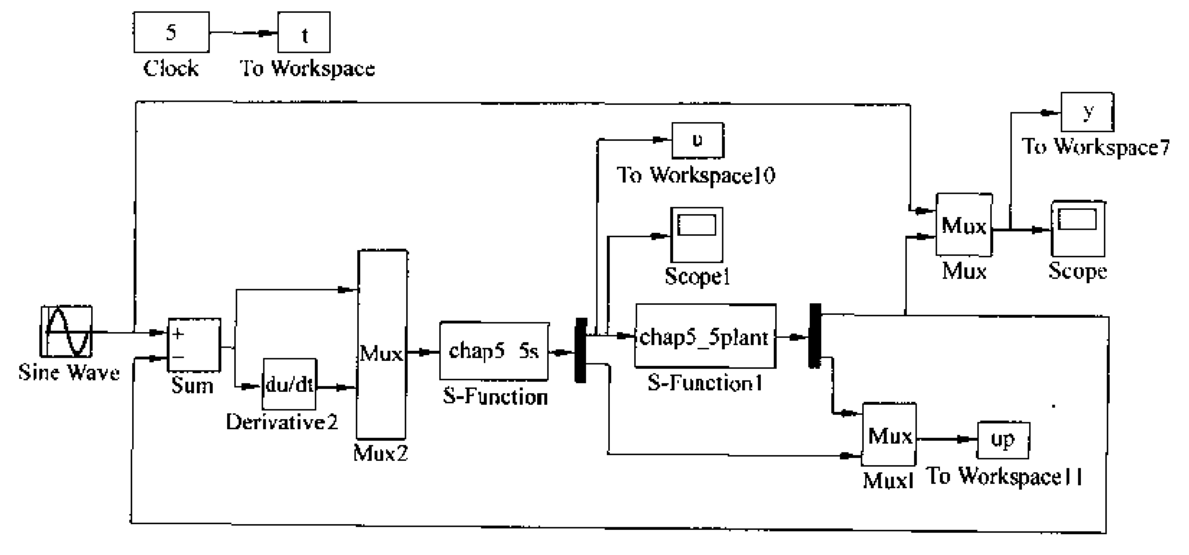


图 5-23　主程序图

(2) 控制器 S 函数：chap5_5s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
global M ci bi c
sizes = simsizes;
sizes.NumContStates = 3;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0.1 * ones(3,1)];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
persistent w m b
if t == 0
    m = [ -0.1693     0.7487   0.5359;
          -0.3900   -0.9700   0.9417];
    b = 0.20 * ones(3,1);
end

c = 30;
m0 = 1/133;

r = sin(pi * t);
dr = pi * cos(pi * t);
ddr = -pi^2 * sin(pi * t);

e = u(1);
de = u(2);
x1 = r + e;
x2 = dr + de;
s = c * e + de;
```

```
xi = [x1;x2];
for j = 1:1:3
    h(j) = exp( - norm(xi - m(:,j))^2/(2 * b(j) * b(j)));
end

eq0 = 0.002;
eq1 = 0.001;
xite = m0^( - 1) * (eq0 - eq1);
for i = 1:1:3
    sys(i) = xite * abs(s * 1/m0) * h(i);
end

function sys = mdlOutputs(t,x,u)
persistent w m b
if t == 0
% m = 1 * rands(2,3)
  m = [ - 0.1693    0.7487   0.5359;
        - 0.3900  - 0.9700   0.9417];
  b = 0.20 * ones(3,1);
end
w = [x(1);x(2);x(3)];

c = 30;
k1 = 100;  % Pole placement
k2 = 200;

r = sin(pi * t);
dr = pi * cos(pi * t);
ddr = - pi^2 * sin(pi * t);

m0 = 1/133;

e = u(1);
de = u(2);
s = c * e + de;

x1 = r + e;
x2 = dr + de;
xi = [x1;x2];
for j = 1:1:3
    h(j) = exp( - norm(xi - m(:,j))^2/(2 * b(j) * b(j)));
end

M = 2;
switch M
case 1  % Konwn UP
  Up = 0.5/133 + 0.10;
case 2  % Using RBF
  Up = w' * h';
end

if s~ = 0
  u0 = - m0 * (c * de + k1 * e + k2 * de) - Up * sign(s);
else
```

```
    u0 = 0;
end
hn = 25/133 * x2;
u1 = m0 * (k1 * e + k2 * de + ddr) + hn;
ut = u1 + u0;

sys(1) = ut;
sys(2) = Up;
```

(3) 被控对象 S 函数:chap5_5plant.m

```
%S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 2;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0.5 0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
dt = 0.5 * sin(2 * pi * t);
sys(1) = x(2);
sys(2) = -25 * x(2) + 133 * u + dt;

function sys = mdlOutputs(t,x,u)
```

```
sys(1) = x(1);
sys(2) = 0.5/133;
```

(4) 作图程序:chap5_5plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,Up(:,1),'r',t,Up(:,2),'b');
xlabel('time(s)');ylabel('Upper bound change');
```

5.5 基于线性化反馈的神经网络滑模控制

利用线性化反馈方法,可设计滑模控制器。采用神经网络逼近及自适应控制方法,利用线性化反馈技术,可设计一种自适应神经滑模控制器[3]。

5.5.1 线性化反馈方法

考虑如下SISO系统:

$$\begin{cases}\dot{\boldsymbol{x}}=f_0(\boldsymbol{x})+g_0(\boldsymbol{x})u\\ y=h(\boldsymbol{x})\end{cases} \tag{5.73}$$

其中 $\boldsymbol{x}\in\mathbf{R}^n$ 为状态变量,$f_0,g_0:\mathbf{R}^n\rightarrow\mathbf{R}^n$,$h:\mathbf{R}^n\rightarrow\mathbf{R}^n$,且 $f_0(0)=0,h(0)=0$。则

$$\dot{y}=\frac{\partial h}{\partial x}\dot{x}=\frac{\partial h}{\partial x}f_0(\boldsymbol{x})+\frac{\partial h}{\partial x}g_0(\boldsymbol{x})u$$
$$:=f_1(\boldsymbol{x})+g_1(\boldsymbol{x})u \tag{5.74}$$

假设 $g_1(x)\neq0$,设计如下线性化反馈控制律:

$$u=\frac{R-f_1(\boldsymbol{x})}{g_1(\boldsymbol{x})} \tag{5.75}$$

则式(5.74)变为线性系统:

$$\dot{y}=R \tag{5.76}$$

设位置指令为 $y_d(t)$,取 R 为

$$R=\dot{y}_d-\alpha(y-y_d) \tag{5.77}$$

其中 $\alpha>0$。则式(5.77)又变为

$$\dot{e}+\alpha e=0 \tag{5.78}$$

其中 $e=y-y_d$。

显然，式(5.78)为误差动态方程，$e(t)$以指数形式趋近于零。如果 $e(0)=\dot{e}(0)=0$，则 $e(t)$在所有时间($t\geqslant 0$)都为零。

5.5.2 滑模控制器的设计

考虑如下 n 阶 SISO 非线性系统：

$$x^{(n)}=f(\boldsymbol{x},t)+g(\boldsymbol{x},t)u \tag{5.79}$$

其中 f 和 g 为未知非线性函数，$\boldsymbol{x}\in\mathbf{R}^n$ 为状态变量。

设位置指令为 $\boldsymbol{x}_d$，则跟踪误差为

$$\boldsymbol{e}=\boldsymbol{x}-\boldsymbol{x}_d=[e \quad \dot{e} \quad \cdots \quad e^{(n-1)}]^T \tag{5.80}$$

定义滑模面为

$$s(\boldsymbol{x},t)=\boldsymbol{ce} \tag{5.81}$$

其中 $\boldsymbol{c}=[c_1 \quad c_2 \quad \cdots \quad c_{n-1} \quad 1]$。

根据线性化反馈技术，将滑模控制律设计为

$$u=\frac{R-f(\boldsymbol{x},t)}{g(\boldsymbol{x},t)} \tag{5.82}$$

$$\xi(\boldsymbol{x},t)=\ddot{x}_d-c\dot{e} \tag{5.83}$$

$$R=\xi(\boldsymbol{x},t)-\rho\,\text{sgn}(s) \quad \rho>0 \tag{5.84}$$

稳定性分析：

定义 Lyapunov 函数

$$V=\frac{1}{2}s^2 \tag{5.85}$$

则

$$\begin{aligned}\dot{V}&=s\dot{s}=s(\ddot{e}+c\dot{e})=s(\ddot{x}-\ddot{x}_d+c\dot{e})\\&=s[f(\boldsymbol{x},t)+g(\boldsymbol{x},t)u-\ddot{x}_d+c\dot{e}]\end{aligned} \tag{5.86}$$

将式(5.82)代入式(5.86)得

$$\dot{V}=s[-\rho\,\text{sgn}(s)] \tag{5.87}$$

即

$$\dot{V}=-\rho|s|\leqslant 0 \tag{5.88}$$

5.5.3 自适应神经滑模控制器的设计

在实际控制中，f 和 g 往往未知，控制律式(5.82)很难实现。可采用 RBF 网络的输出 $\hat{f}(\boldsymbol{x},t)$和 $\hat{g}(\boldsymbol{x},t)$代替 $f(\boldsymbol{x},t)$和 $g(\boldsymbol{x},t)$，实现自适应神经滑模控制。

采用 RBF 网络建立$\hat{f}(\boldsymbol{x},t)$和 $\hat{g}(\boldsymbol{x},t)$的模型，网络结构如图 5-24 所示。

1. 神经滑模控制器的设计

假设存在权值 $\boldsymbol{w}$ 和 $\boldsymbol{v}$，使得$\hat{f}$和 $\hat{g}$ 逼近于 f 和 g，其逼近精度为 ε，即

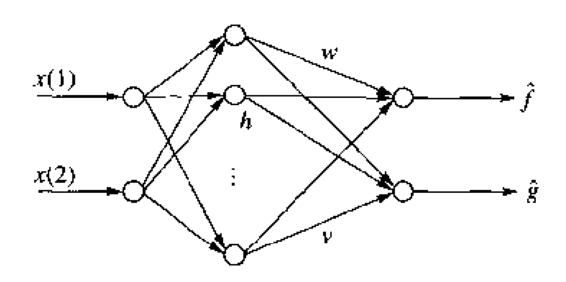


图 5-24　RBF 网络结构

$$\max\|\hat{f}(\boldsymbol{x},\boldsymbol{w})-f(\boldsymbol{x})\|\leqslant\varepsilon,\max\|\hat{g}(\boldsymbol{x},\boldsymbol{v})-g(\boldsymbol{x})\|\leqslant\varepsilon \tag{5.89}$$

假设 $\boldsymbol{w}_t$ 和 $\boldsymbol{v}_t$ 分别表示 $\boldsymbol{w}$ 和 $\boldsymbol{v}$ 在 t 时刻的估计值，并且 $\hat{g}(\boldsymbol{x},\boldsymbol{v}_t)$ 在自适应过程中是非奇异的。则控制律式(5.82)变为

$$u(t)=\frac{R-\hat{f}(\boldsymbol{x},\boldsymbol{w}_t)}{\hat{g}(\boldsymbol{x},\boldsymbol{v}_t)} \tag{5.90}$$

$$\hat{f}(\boldsymbol{x},\boldsymbol{w}_t)=\boldsymbol{w}_t\boldsymbol{h},\hat{g}(\boldsymbol{x},\boldsymbol{v}_t)=\boldsymbol{v}_t\boldsymbol{h} \tag{5.91}$$

2. 稳定性分析

针对二阶系统，$n=2$，则

$$\begin{aligned}\dot{s}&=c\dot{e}+\ddot{e}=c\dot{e}+\ddot{x}-\ddot{x}_{\mathrm{d}}=c\dot{e}+f(\boldsymbol{x},t)+g(\boldsymbol{x},t)u-\ddot{x}_{\mathrm{d}}\\&=f(\boldsymbol{x},t)+g(\boldsymbol{x},t)u-\xi(\boldsymbol{x},t)\end{aligned}$$

将控制律 u 代入上式，得

$$\begin{aligned}\dot{s}&=f(\boldsymbol{x},t)+[g(\boldsymbol{x},t)+\hat{g}(\boldsymbol{x},\boldsymbol{v}_t)-\hat{g}(\boldsymbol{x},\boldsymbol{v}_t)]u-\xi(\boldsymbol{x},t)\\&=f(\boldsymbol{x},t)+[g(\boldsymbol{x},t)-\hat{g}(\boldsymbol{x},\boldsymbol{v}_t)]u+\hat{g}(\boldsymbol{x},\boldsymbol{v}_t)[\hat{g}^{-1}(\boldsymbol{x},\boldsymbol{v}_t)(-\hat{f}(\boldsymbol{x},\boldsymbol{w}_t)+R)]-\xi(\boldsymbol{x},t)\\&=[f(\boldsymbol{x},t)-\hat{f}(\boldsymbol{x},\boldsymbol{w}_t)]+[g(\boldsymbol{x},t)-\hat{g}(\boldsymbol{x},\boldsymbol{v}_t)]u+R-\xi(\boldsymbol{x},t)\\&=[f(\boldsymbol{x},t)-\hat{f}(\boldsymbol{x},\boldsymbol{w}_t)]+[g(\boldsymbol{x},t)-\hat{g}(\boldsymbol{x},\boldsymbol{v}_t)]u+\xi(\boldsymbol{x},t)-\rho\operatorname{sgn}(s)-\xi(\boldsymbol{x},t)\\&=[f(\boldsymbol{x},t)-\hat{f}(\boldsymbol{x},\boldsymbol{w}_t)]+[g(\boldsymbol{x},t)-\hat{g}(\boldsymbol{x},\boldsymbol{v}_t)]u-\rho\operatorname{sgn}(s)\\&=[(\hat{f}(\boldsymbol{x},\boldsymbol{w})-\hat{f}(\boldsymbol{x},\boldsymbol{w}_t))+(\hat{g}(\boldsymbol{x},\boldsymbol{v})-\hat{g}(\boldsymbol{x},\boldsymbol{v}_t))u]+[(f(\boldsymbol{x},t)-\hat{f}(\boldsymbol{x},\boldsymbol{w}))+(g(\boldsymbol{x},t)\\&\quad-\hat{g}(\boldsymbol{x},\boldsymbol{v}))u]-\rho\operatorname{sgn}(s)\\&=-\frac{\partial\hat{f}(\boldsymbol{x},\boldsymbol{w})}{\partial\boldsymbol{w}}\bigg|_{\boldsymbol{w}_t}\boldsymbol{w}_t-\frac{\partial\hat{g}(\boldsymbol{x},\boldsymbol{v})}{\partial\boldsymbol{v}}\bigg|_{\boldsymbol{v}_t}\boldsymbol{v}_t u+\frac{\partial\hat{f}(\boldsymbol{x},\boldsymbol{w})}{\partial\boldsymbol{w}}\bigg|_{\boldsymbol{w}}\boldsymbol{w}+\frac{\partial\hat{g}(\boldsymbol{x},\boldsymbol{v})}{\partial\boldsymbol{v}}\bigg|_{\boldsymbol{v}}\boldsymbol{v}u-\rho\operatorname{sgn}(s)\end{aligned}$$

定义

$$\boldsymbol{\Theta}_t=[\boldsymbol{w}_t\quad\boldsymbol{v}_t]^{\mathrm{T}} \tag{5.92}$$

及

$$\widetilde{\boldsymbol{\Theta}}(t)=\boldsymbol{\Theta}_t-\boldsymbol{\Theta} \tag{5.93}$$

其中 $\boldsymbol{\Theta}=[\boldsymbol{w}\quad\boldsymbol{v}]^{\mathrm{T}}$，则

$$\dot{s}=[-\widetilde{\boldsymbol{\Theta}}^{\mathrm{T}}\boldsymbol{J}+\eta(t)]-\rho\operatorname{sgn}(s)$$

其中

$$J=\left[\left(\frac{\partial \hat{f}(\boldsymbol{x},\boldsymbol{w})}{\partial \boldsymbol{w}}\bigg|_{w_t}\right) \quad \left(\frac{\partial \hat{g}(\boldsymbol{x},\boldsymbol{v})}{\partial \boldsymbol{v}}\bigg|_{v_t}\right)u\right]^{\mathrm{T}}=[\boldsymbol{h} \quad \boldsymbol{h}u]^{\mathrm{T}} \tag{5.94}$$

$$\eta(t)=O(\|\tilde{\boldsymbol{\Theta}}\|^2)+O(\varepsilon)$$

按下式调整网络权值：

$$\dot{\boldsymbol{\Theta}}_t=\boldsymbol{J}s \tag{5.95}$$

即

$$\dot{\tilde{\boldsymbol{\Theta}}}(t)=\boldsymbol{J}s \tag{5.96}$$

选择滑模切换增益为

$$\rho>|\eta(t)| \tag{5.97}$$

定义 Lyapunov 函数

$$V(s,\boldsymbol{\Theta})=\frac{1}{2}s^2+\frac{1}{2}\tilde{\boldsymbol{\Theta}}(t)^{\mathrm{T}}\tilde{\boldsymbol{\Theta}}(t) \tag{5.98}$$

则

$$\begin{aligned}\dot{V}&=s\dot{s}+\tilde{\boldsymbol{\Theta}}(t)^{\mathrm{T}}\dot{\tilde{\boldsymbol{\Theta}}}(t)=-\rho|s|+s[-\tilde{\boldsymbol{\Theta}}(t)^{\mathrm{T}}\boldsymbol{J}+\eta(t)]+\tilde{\boldsymbol{\Theta}}(t)^{\mathrm{T}}\dot{\tilde{\boldsymbol{\Theta}}}(t)\\&=-\rho|s|+s\eta(t)\leqslant-\rho|s|+|\eta(t)||s|=-(\rho-|\eta(t)|)|s|\leqslant 0\end{aligned} \tag{5.99}$$

为了降低抖振，采用连续函数 s_δ 代替 sgn(s)：

$$s_\delta=\frac{s}{|s|+\delta} \tag{5.100}$$

$$\delta=\delta_0+\delta_1\|\boldsymbol{e}\| \tag{5.101}$$

其中 δ_0，δ_1 为两个正常数。

5.5.4 仿真实例

被控对象取单级倒立摆，其动态方程如下：

$$\dot{x}_1=x_2$$

$$\dot{x}_2=\frac{g\sin x_1-mlx_2^2\cos x_1\sin x_1/(m_c+m)}{l[4/3-m\cos^2 x_1/(m_c+m)]}+\frac{\cos x_1/(m_c+m)}{l[4/3-m\cos^2 x_1/(m_c+m)]}u$$

其中 x_1 和 x_2 分别为摆角和摆速；$g=9.8\text{m/s}^2$；m_c 为小车质量，$m_c=1\text{kg}$；m 为摆杆质量，$m=0.1\text{kg}$；l 为摆长的 1/2，$l=0.5\text{m}$；u 为控制输入。

位置指令为 $x_d(t)=0.1\sin(\pi t)$，取切换函数为 $s=ce+\dot{e}$，$c=10$。设 $\boldsymbol{w}$ 和 $\boldsymbol{v}$ 的初始值均为[0.1 0.1 0.1 0.1]，高斯参数 $\boldsymbol{c}_i$ 和 $\boldsymbol{b}_i$ 的初始值分别为 $\begin{bmatrix}10 & 10 & 10 & 10\\10 & 10 & 10 & 10\end{bmatrix}$ 和 $[10 \quad 10 \quad 10 \quad 10]^{\mathrm{T}}$。

采用控制律式(5.90)，倒立摆初始状态为$[-\pi/60 \quad 0]$。在程序中，$M=1$ 为采用符号函数，$M=2$ 为采用连续函数式(5.100)。取 $M=2$，$\delta_0=0.03$，$\delta_1=5$，$\delta=\delta_0+\delta_1|e|$，$k=5$。仿真结果如图 5-25～图 5-28 所示。

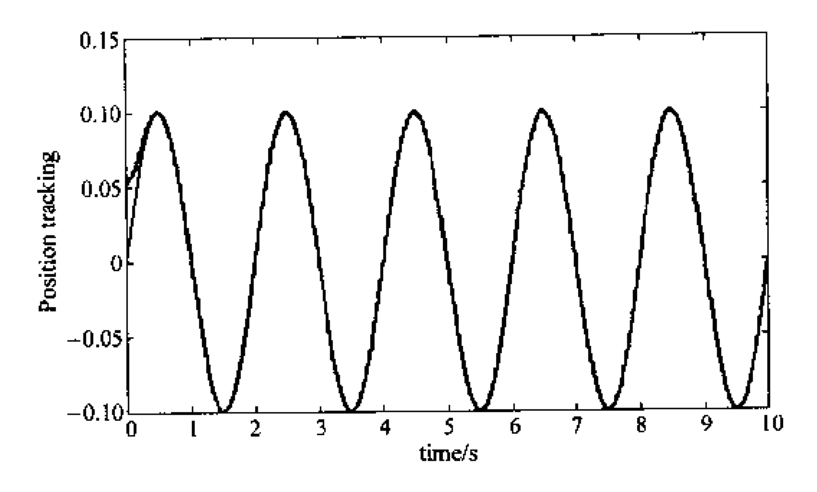


图 5-25 位置跟踪

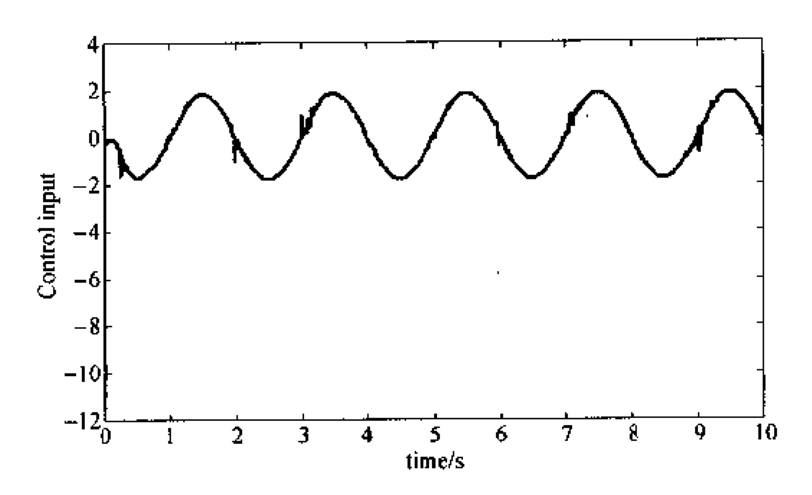


图 5-26 控制输入信号

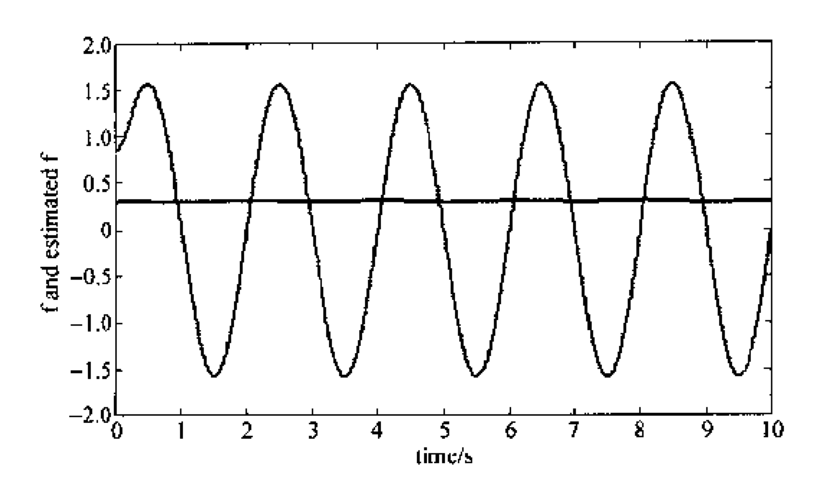


图 5-27 $f(x,t)$及$\hat{f}(x,t)$的变化

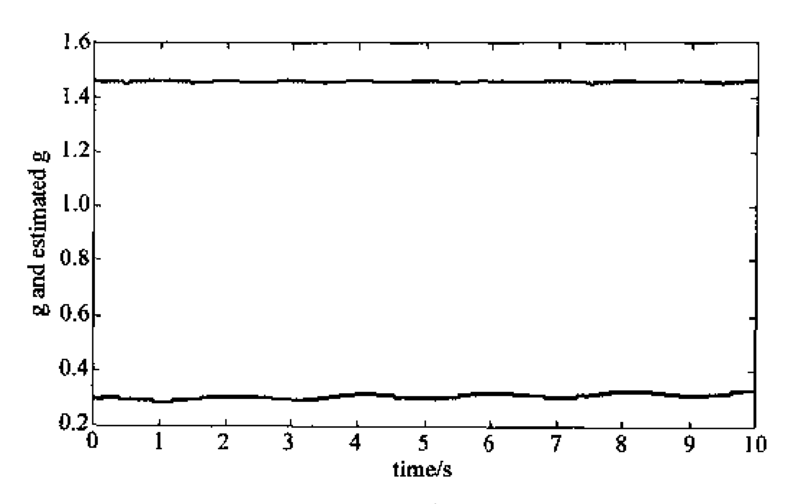


图 5-28 $g(x,t)$及$\hat{g}(x,t)$的变化

仿真程序：

(1) Simulink 主程序(如图 5-29 所示)：chap5_6sim. mdl

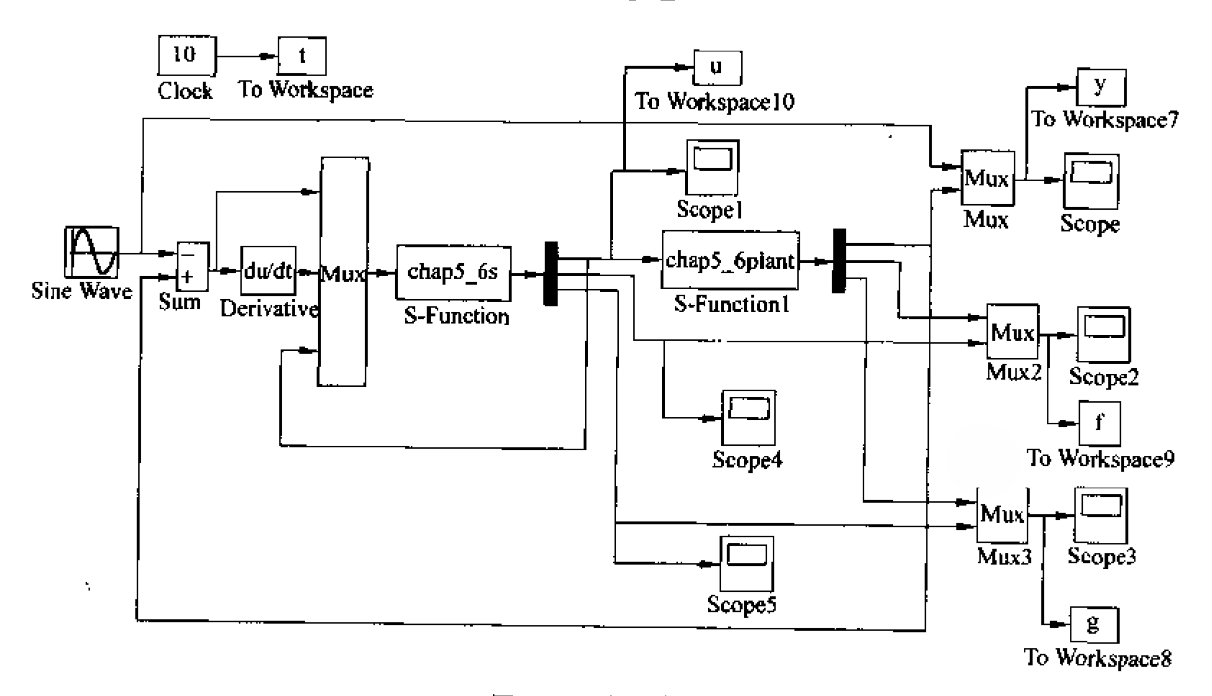


图 5-29 主程序图

(2) 控制器 S 函数：chap5_6s. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
```

```
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 8;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [0.2 * ones(8,1)];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)

r = 0.1 * sin(pi * t);
dr = 0.1 * pi * cos(pi * t);
ddr = - 0.1 * pi * pi * sin(pi * t);

e = u(1);
de = u(2);

c = 10;
s = de + c * e;

w = 0.1 * ones(1,4);
v = 0.1 * ones(1,4);
bi = 10 * ones(4,1);
ci = 10 * ones(2,4);

xx(1) = r + e;
xx(2) = dr + de;

for j = 1:1:4
    h(j) = exp( - norm(xx' - ci(:,j))^2/(2 * bi(j) * bi(j)));
end
fx1 = w * h';
gx1 = v * h';

w = [x(1),x(2),x(3),x(4)];
v = [x(5),x(6),x(7),x(8)];

J1 = h;
J2 = h * u(3);

for i = 1:1:4
    sys(i) = J1(i) * s;
end
```

```
for i = 5:1:8
    sys(i) = J2(i-4) * s;
end

function sys = mdlOutputs(t,x,u)

r = 0.1 * sin(pi * t);
dr = 0.1 * pi * cos(pi * t);
ddr = -0.1 * pi * pi * sin(pi * t);

e = u(1);
de = u(2);

c = 10;
s = de + c * e;
kesi = ddr - c * de;

M = 2;
if M == 1
   K = 1.0;
   R = kesi - K * sign(s);
elseif M == 2
   K = 5;
   delta0 = 0.03;
   delta1 = 5;
   delta = delta0 + delta1 * abs(e);
   R = kesi - K * s/(abs(s) + delta);
end

w = 0.1 * ones(1,4);
v = 0.1 * ones(1,4);
bi = 10 * ones(4,1);
ci = 10 * ones(2,4);

xx(1) = r + e;
xx(2) = dr + de;

for j = 1:1:4
    h(j) = exp(-norm(xx'-ci(:,j))^2/(2 * bi(j) * bi(j)));
end
w = [x(1),x(2),x(3),x(4)];
v = [x(5),x(6),x(7),x(8)];

fx1 = w * h';
gx1 = v * h';

ut = (-fx1 + R)/(gx1 + 0.001);

sys(1) = ut;
sys(2) = fx1;
sys(3) = gx1;
```

(3) 被控对象 S 函数：chap5_6plant.m

```
%S-function for continuous state equation
function [sys,x0,str,ts]=s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys=mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys=simsizes(sizes);
x0=[pi/60 0];
str=[];
ts=[];

function sys=mdlDerivatives(t,x,u)

g=9.8;
mc=1.0;
m=0.1;
l=0.5;

S=l*(4/3-m*(cos(x(1)))^2/(mc+m));
fx=g*sin(x(1))-m*l*x(2)^2*cos(x(1))*sin(x(1))/(mc+m);
fx=fx/S;
gx=cos(x(1))/(mc+m);
gx=gx/S;

%fx=-25*x(2);
```

```
%gx = 133;

sys(1) = x(2);
sys(2) = fx + gx * u;

function sys = mdlOutputs(t,x,u)

g = 9.8;
mc = 1.0;
m = 0.1;
l = 0.5;

S = l * (4/3 - m * (cos(x(1)))^2/(mc + m));
fx = g * sin(x(1)) - m * l * x(2)^2 * cos(x(1)) * sin(x(1))/(mc + m);
fx = fx/S;
gx = cos(x(1))/(mc + m);
gx = gx/S;

%fx = -25 * x(2);
%gx = 133;

sys(1) = x(1);
sys(2) = fx;
sys(3) = gx;
```

(4) 作图程序:chap5_6plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,f(:,1),'r',t,f(:,2),'b');
xlabel('time(s)');ylabel('f and estimated f');

figure(4);
plot(t,g(:,1),'r',t,g(:,2),'b');
xlabel('time(s)');ylabel('g and estimated g');
```

5.6 基于 RBF 网络切换增益调节的滑模控制

滑模控制的抖振大小是由其控制器切换项的增益决定的,采用神经网络对切换项的增益进行调节,可降低滑模控制的抖振[4]。

5.6.1　系统描述

考虑如下不确定系统：

$$\ddot{\theta}(t)=f(\boldsymbol{x},t)+g(\boldsymbol{x},t)u(t)+d(t) \tag{5.102}$$

其中 $\boldsymbol{x}=[\theta\quad \dot{\theta}]^{\mathrm{T}}$，$d(t)$为外部干扰。

假设系统满足：

$$0<g_{\min}\leqslant g(\boldsymbol{x},t)\leqslant g_{\max} \tag{5.103}$$

$$|f-\hat{f}|\leqslant F(\boldsymbol{x},t) \tag{5.104}$$

$$\frac{1}{\alpha}\leqslant\frac{\hat{g}}{g}\leqslant\alpha \tag{5.105}$$

$$\alpha=\left(\frac{g_{\max}}{g_{\min}}\right)^{\frac{1}{2}}+1\geqslant 2 \tag{5.106}$$

$$|d|\leqslant D(\boldsymbol{x},t) \tag{5.107}$$

其中$\hat{f}$和 $\hat{g}$ 分别为 $f(\boldsymbol{x},t)$和 $g(\boldsymbol{x},t)$的名义值。

5.6.2　固定增益滑模控制器的设计

定义跟踪误差为

$$e=\theta-\theta_{\mathrm{d}} \tag{5.108}$$

设计滑模面为

$$s(\boldsymbol{x},t)=ce+\dot{e} \tag{5.109}$$

其中 $c>0$。则

$$\dot{s}(\boldsymbol{x},t)=c\dot{e}+\ddot{e}=f+gu+d-\ddot{\theta}_{\mathrm{d}}+c\dot{e} \tag{5.110}$$

令 $\dot{s}(\boldsymbol{x},t)=0$，并假设不确定性和干扰为零，得

$$\hat{f}+\hat{g}u_{\mathrm{eq}}-\ddot{\theta}_{\mathrm{d}}+c\dot{e}=0 \tag{5.111}$$

从而得到等效控制器

$$u_{\mathrm{eq}}=\hat{g}^{-1}\hat{u} \tag{5.112}$$

其中

$$\hat{u}=-\hat{f}+\ddot{\theta}_{\mathrm{d}}-c\dot{e} \tag{5.113}$$

设计切换控制器为

$$u_{\mathrm{n}}=-\hat{g}^{-1}K\mathrm{sgn}(s) \tag{5.114}$$

其中 K 为增益项，$\eta>0$，$|\ddot{\theta}_{\mathrm{d}}|\leqslant W$。

增益项 K 设计为

$$K=\alpha[(F+D+W+\eta)+(\alpha-1)|\hat{u}|] \tag{5.115}$$

总控制器为

$$u=u_{\mathrm{eq}}+u_{\mathrm{n}} \tag{5.116}$$

稳定性分析：

定义 Lyapunov 函数为

$$V=\frac{1}{2}s^2 \tag{5.117}$$

则

$$\begin{aligned}\dot{V}&=s\dot{s}=s(f+gu+d-\ddot{\theta}_d+c\dot{e})=s[f+g(-\hat{g}^{-1}K\text{sgn}(s)+\hat{g}^{-1}\hat{u})+d-\ddot{\theta}_d+c\dot{e}]\\&=s[f-g\hat{g}^{-1}K\text{sgn}(s)+g\hat{g}^{-1}\hat{u}+d-\ddot{\theta}_d+c\dot{e}]\end{aligned}$$

由式(5.113)可知

$$c\dot{e}=-\hat{f}+\ddot{\theta}_d-\hat{u} \tag{5.118}$$

则

$$\begin{aligned}\dot{V}&=s(f-\hat{f}-\hat{u}+g\hat{g}^{-1}\hat{u}-g\hat{g}^{-1}K\text{sgn}(s)+d)\\&=s[(f-\hat{f}+d)+(g\hat{g}^{-1}-1)\hat{u}-g\hat{g}^{-1}K\text{sgn}(s)]\\&=s(f-\hat{f}+d)+s(g\hat{g}^{-1}-1)\hat{u}-g\hat{g}^{-1}K|s|\end{aligned}$$

由式(5.105)得

$$\frac{1}{\alpha}\leqslant\frac{g}{\hat{g}}\leqslant\alpha \tag{5.119}$$

则

$$\begin{aligned}\dot{V}&\leqslant s(f-\hat{f})+sd+s(g\hat{g}^{-1}-1)\hat{u}-\frac{1}{\alpha}K|s|\\&\leqslant|s|(F+D)+|s||g\hat{g}^{-1}-1||\hat{u}|-\frac{1}{\alpha}K|s|\end{aligned}$$

将式(5.115)中的 K 代入，得

$$\begin{aligned}\dot{V}&\leqslant(F+D)|s|+|s||g\hat{g}^{-1}-1||\hat{u}|-[F+D+W+\eta+(\alpha-1)|\hat{u}|]|s|\\&=|s||g\hat{g}^{-1}-1||\hat{u}|-[W+\eta+(\alpha-1)|\hat{u}|]|s|\end{aligned}$$

由式(5.105)和式(5.106)得

$$\alpha-1\geqslant1,-\frac{1}{2}\leqslant g\hat{g}^{-1}-1\leqslant\alpha-1$$

则

$$|g\hat{g}^{-1}-1|\leqslant\alpha-1 \tag{5.120}$$

$$\dot{V}\leqslant-(W+\eta)|s|\leqslant-\eta|s| \tag{5.121}$$

5.6.3 基于 RBF 网络的增益调节

采用 RBF 神经网络来调节切换项的增益 K。设 RBF 网络的输入为 $\boldsymbol{x}=[s\ \dot{s}]$，输出的绝对值为切换项的增益 K。

取

$$K=|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})| \tag{5.122}$$

其中 $\boldsymbol{w}^{\mathrm{T}}$ 为 RBF 神经网络的权值，$\boldsymbol{h}(\boldsymbol{x})$为高斯函数。

$$h_i(\boldsymbol{x})=\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{m}_i\|^2}{\sigma_i^2}\right)\quad i=1,2,3;j=1,2 \tag{5.123}$$

其中 m_i 是第 i 个神经元的中心位置，σ 为第 i 个神经元的宽度。

神经网络权值调整的指标为

$$E=\frac{1}{2}e^2 \tag{5.124}$$

其中 $e=y-r$，r 为阶跃响应信号。

网络权值的学习算法为

$$\begin{aligned}\Delta w&=-\eta_1\frac{\partial E}{\partial w}=-\eta_1 e\frac{\partial e}{\partial w}=-\eta_1 e\frac{\partial y}{\partial w}=-\eta_1 e\frac{\partial y}{\partial u}\frac{\partial u}{\partial K}\frac{\partial K}{\partial w}\\&\approx-\eta_1 e\,\mathrm{sgn}\left(\frac{\partial y}{\partial u}\right)\frac{\partial u}{\partial K}\frac{\partial K}{\partial w}\end{aligned} \tag{5.125}$$

分以下三种情况讨论：

(1) $\frac{\partial y}{\partial u}$主要取决于正负号，其值的大小可以通过权值来补偿。在阶跃响应过程中，y 的值正比于 u，故 $\mathrm{sgn}\left(\frac{\partial y}{\partial u}\right)=1$。

(2) $\frac{\partial u}{\partial K}=\frac{\partial u_n}{\partial K}=-\hat{g}^{-1}\mathrm{sgn}(s)$。

(3) $\frac{\partial K}{\partial w}=h(x)\mathrm{sgn}(w^{\mathrm{T}}h(x))$。

则权值调整算法为

$$\Delta w(t)\approx-\eta_1 e(-\hat{g}^{-1})\mathrm{sgn}(s)h(x)\mathrm{sgn}(w^{\mathrm{T}}h(x)) \tag{5.126}$$

网络权值学习算法为

$$w(t)=w(t-1)+\Delta w(t)+\alpha(w(t)-w(t-1)) \tag{5.127}$$

式中，η_1 为网络学习速率，$\eta_1\in(0,1)$，α 为惯性量系数，$\alpha\in(0,1)$。

5.6.4　仿真实例

设被控对象为

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=-[25+5\sin(t)]x_2+[133+15\sin(t)]u+d(t)\end{cases}$$

其中 $d(t)=50\exp\left(-\frac{(t-c)^2}{2b^2}\right)$，$b=0.05$，$c=2$。

阶跃响应取 $r=1.0$。在控制律式(5.126)中，取 $\eta=0.5$，$W=0$。程序中，$M=1$ 为采用固定增益的控制器，$M=2$ 为采用 RBF 进行增益调整的控制器。取 $M=1$，仿真结果如图 5-30 和图 5-31 所示。取 $M=2$，采用神经网络调节 K，网络初始权值取 $w(0)=[30\quad 30\quad 30]^{\mathrm{T}}$，高斯参数取 $m=\begin{bmatrix}3&3&3\\3&3&3\end{bmatrix}$，$\sigma=[5\quad 5\quad 5]^{\mathrm{T}}$。网络学习参数取 $\eta_1=0.80$，$\alpha_1=0.05$，仿真结果如图 5-32～图 5-34 所示。由仿真可见，采用神经网络进行切换项增益 K 的调节，可大大地降低抖振。

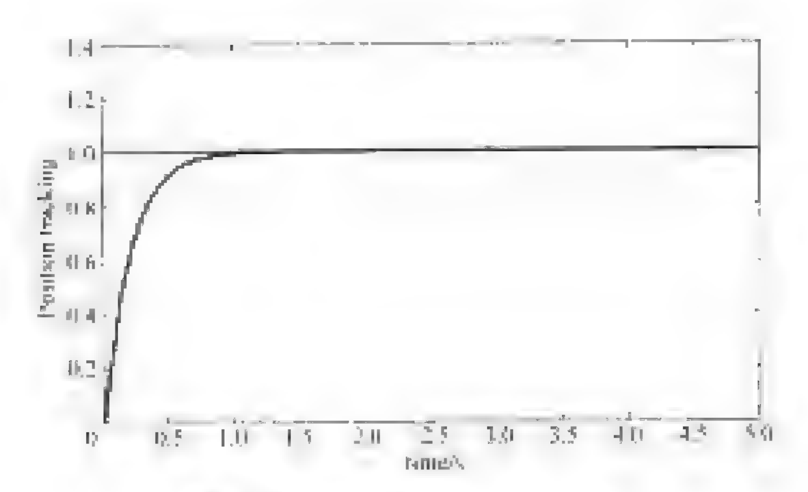


图 5.30 固定增益阶跃响应(M=1)

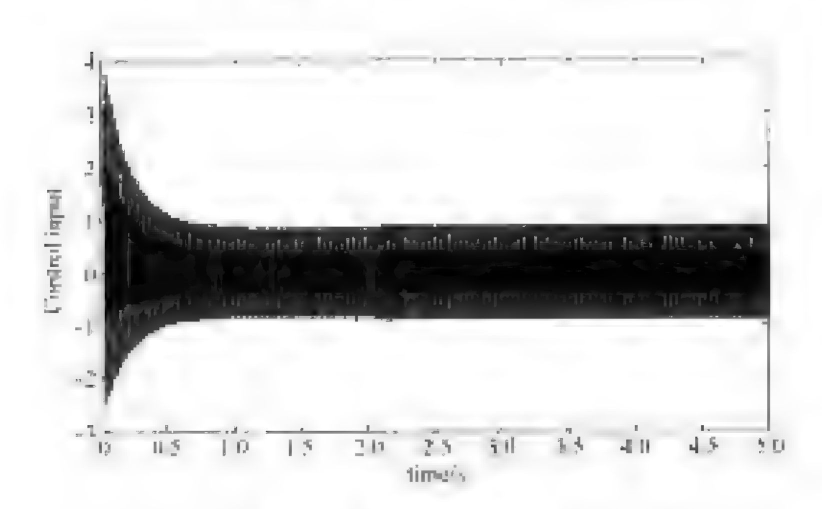


图 5.31 控制输入(M=1)

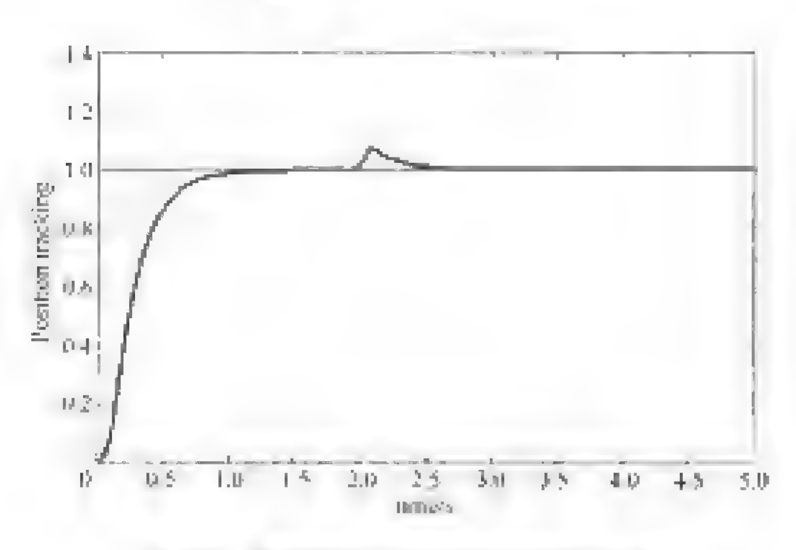


图 5.32 神经网络增益调节阶跃响应(M=2)

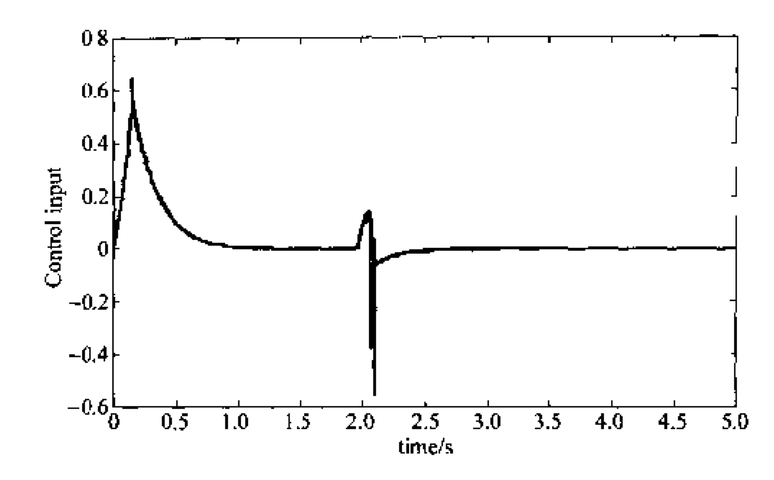


图 5-33　控制输入($M=2$)

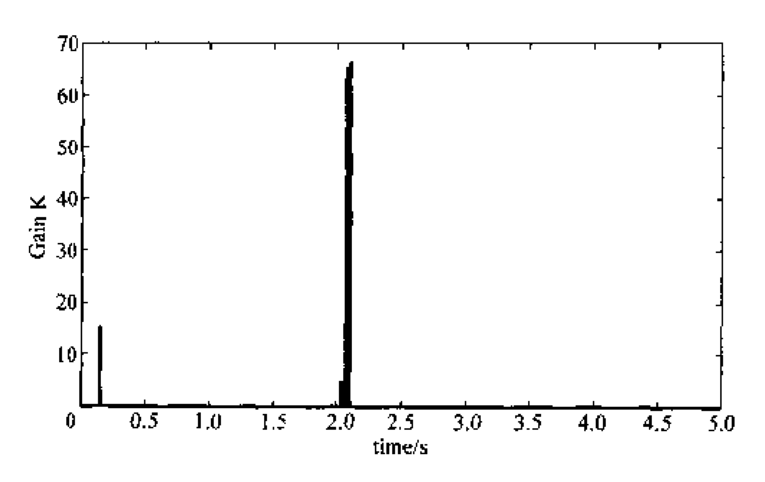


图 5-34　增益变化($M=2$)

仿真程序：

(1) Simulink 主程序(如图 5-35 所示)：chap5_7sim. mdl

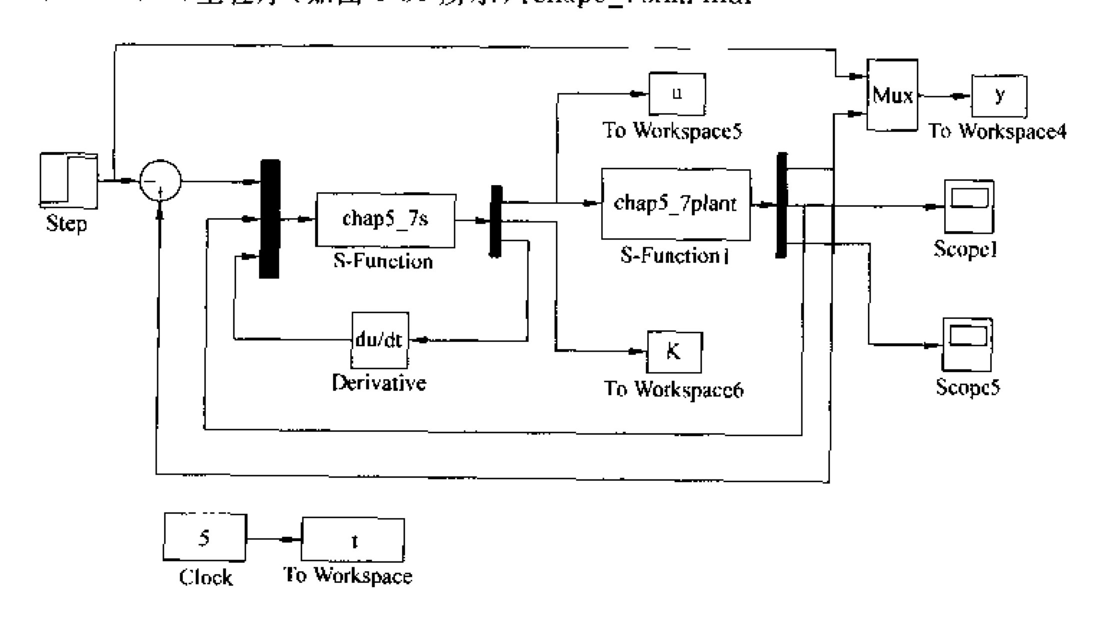


图 5-35　主程序图

(2) 控制器 S 函数:chap5_7s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)
persistent w w_1 w_2 m b

Gp = 133;
Gmax = 150;
Gmin = 100;
alfa = sqrt(Gmax/Gmin) + 1;

r = 1;dr = 0;ddr = 0;

e = u(1);
de = u(2) - dr;
c = 5;
s = c * e + de;

fp = - 25 * u(2);
F = 5 * abs(u(2));

D = 50;

Up = - fp - c * de + ddr;
ueq = 1/Gp * Up;
```

```
xite = 0.50;

M = 2;
switch M
case 1
    K = alfa * (F + D + xite + (alfa - 1) * abs(Up));
case 2 % Using RBF

if t == 0
   w = 30 * ones(3,1);
   m = 3 * ones(2,3);
   b = 5 * ones(3,1);
   w_1 = w;
   w_2 = w_1;
end

    ds = u(3);
    xi = [s;ds];
for j = 1:1:3
    h(j) = exp( - norm(xi - m(:,j))^2/(2 * b(j) * b(j)));
end
    ym = w' * h';
    K = abs(ym)
    dyu = 1.0;

    xite1 = 0.80;
    alfa1 = 0.05;
    d_w = xite1 * e * dyu * 1/Gp * sign(s) * h' * sign(ym);
    w = w_1 + d_w + alfa1 * (w_1 - w_2);
end
un = - 1/Gp * K * sign(s);
ut = ueq + un;

sys(1) = ut;
sys(2) = K;
sys(3) = s;
```

(3) 被控对象 S 函数:chap5_7plant.m

```
% S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
```

```
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 2;
sizes.NumDiscStates = 0;
sizes.NumOutputs = 3;
sizes.NumInputs = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)

fp = -25 * x(2);
f = fp - 5 * sin(t) * x(2);
Gp = 133;
G = Gp + 15 * sin(t);

b = 0.05;c = 2;
dt = 50 * exp( - (t - c)^2/(2 * b^2)); % rbf_func.m

sys(1) = x(2);
sys(2) = f + G * u + dt;
function sys = mdlOutputs(t,x,u)
b = 0.05;c = 2;
dt = 50 * exp( - (t - c)^2/(2 * b^2)); % rbf_func.m

sys(1) = x(1);
sys(2) = x(2);
sys(3) = dt;
```

(4) 作图程序:chap5_7plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('position tracking');
```

```
figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,K(:,1),'r');
xlabel('time(s)');ylabel('Gain K');
```

参考文献

1. Huang S J, Huang K S, Chiou K C. Development and application of a novel radial basis function sliding mode controller. Mechatronics, 2003,13:313～329
2. Man Z H, Yu X H, Eshraghian K, Palaniswami. M. A robust adaptive sliding mode tracking control using an RBF neural network for robotic manipulators. IEEE International Conference on Neural Networks, 1995, 5: 2403～2408
3. Horng J H. Neural adaptive tracking control of a DC motor. Information Sciences, 1999,118:1～13
4. Lin F J, Wai R J. Sliding-mode-controlled slider-crank mechanism with fuzzy neural network. IEEE Transactions on Industrial Electronics, 2001, 48(1): 60～70
5. 刘金琨. 智能控制. 北京:电子工业出版社, 2005

第6章　基于反演设计的滑模控制

6.1　一种简单反演控制器的设计

6.1.1　基本原理

反演(backstepping)设计方法的基本思想是将复杂的非线形系统分解成不超过系统阶数的子系统，然后为每个子系统分别设计李雅普诺夫函数和中间虚拟控制量，一直"后退"到整个系统，直到完成整个控制律的设计。

反演设计方法，又称反步法、回推法或后推法，通常与李雅普诺夫型自适应律结合使用，综合考虑控制律和自适应律，使整个闭环系统满足期望的动静态性能指标。

假设被控对象为

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(x,t) + b(x,t)u \end{cases} \tag{6.1}$$

其中 $b(\boldsymbol{x},t) \neq 0$。

基本的 backstepping 控制方法设计步骤如下：

步骤一：

定义位置误差

$$z_1 = x_1 - z_d \tag{6.2}$$

其中 z_d 为指令信号。则

$$\dot{z}_1 = \dot{x}_1 - \dot{z}_d = x_2 - \dot{z}_d \tag{6.3}$$

定义虚拟控制量

$$\alpha_1 = -c_1 z_1 + \dot{z}_d \tag{6.4}$$

其中 $c_1 > 0$。

定义

$$z_2 = x_2 - \alpha_1 \tag{6.5}$$

定义 Lyapunov 函数

$$V_1 = \frac{1}{2} z_1^2 \tag{6.6}$$

则

$$\dot{V}_1 = z_1 \dot{z}_1 = z_1 (x_2 - \dot{z}_d) = z_1 (z_2 + \alpha_1 - \dot{z}_d) \tag{6.7}$$

将式(6.4)代入式(6.7)得

$$\dot{V}_1 = -c_1 z_1^2 + z_1 z_2 \tag{6.8}$$

如果 $z_2=0$，则 $\dot{V}_1\leqslant 0$。为此，需要进行下一步设计。

步骤二：

定义 Lyapunov 函数

$$V_2 = V_1 + \frac{1}{2}z_2^2 \tag{6.9}$$

由于

$$\dot{z}_2 = \dot{x}_2 - \dot{\alpha}_1 = f(\boldsymbol{x},t) + b(\boldsymbol{x},t)u + c_1\dot{z}_1 - \ddot{z}_d$$

则

$$\dot{V}_2 = \dot{V}_1 + z_2\dot{z}_2 = -c_1z_1^2 + z_1z_2 + z_2[f(\boldsymbol{x},t) + b(\boldsymbol{x},t)u + c_1\dot{z}_1 - \ddot{z}_d]$$

为使 $\dot{V}_2\leqslant 0$，设计控制器为

$$u = \frac{1}{b(\boldsymbol{x},t)}[-f(\boldsymbol{x},t) - c_2z_2 - z_1 - c_1\dot{z}_1 + \ddot{z}_d] \tag{6.10}$$

其中 c_2 为大于零的正常数。则

$$\dot{V}_2 = -c_1z_1^2 - c_2z_2^2 \leqslant 0 \tag{6.11}$$

通过控制律的设计，使得系统满足了李雅普诺夫稳定性理论条件，z_1 和 z_2 以指数形式渐进稳定，从而保证系统具有全局意义下指数的渐进稳定性。

6.1.2 仿真实例

被控对象为

$$\begin{cases}\dot{x}_1 = x_2 \\ \dot{x}_2 = -25x_2 + 133u\end{cases} \tag{6.12}$$

其中 $f(\boldsymbol{x},t)=-25x_2$，$b(\boldsymbol{x},t)=133$。

以正弦信号为指令信号，初始状态为 $\boldsymbol{x}(0)=[0.5\ 0]$，指令信号设为 $r=\sin(6\pi t)$。控制器参数取 $c_1=35$，$c_2=15$，仿真结果如图 6-1 和图 6-2 所示。

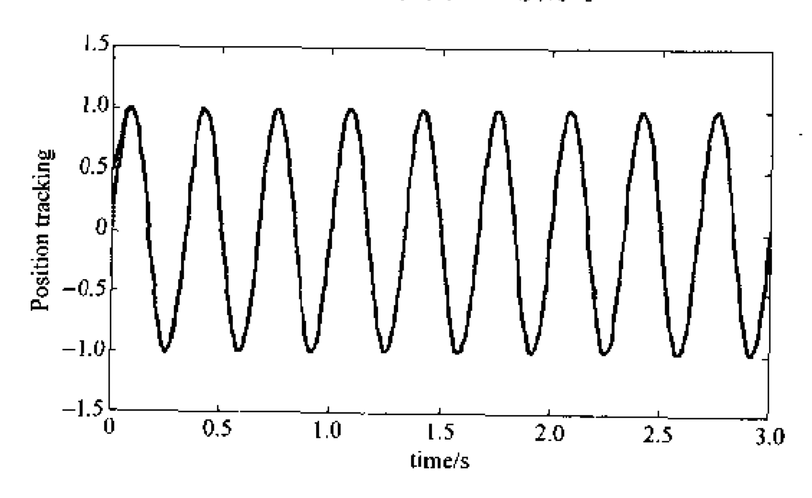


图 6-1　正弦位置跟踪

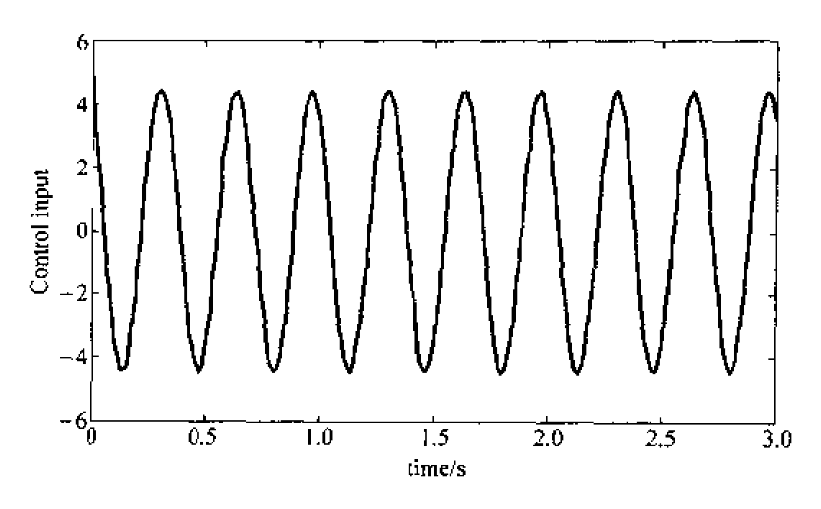


图 6-2 控制输入

仿真程序：

（1）Simulink 主程序（如图 6-3 所示）：chap6_1sim. mdl

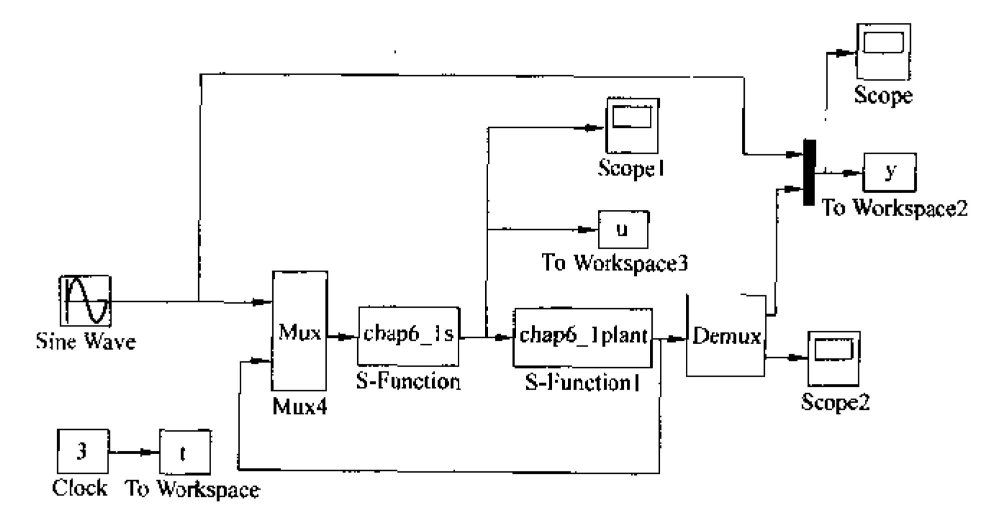


图 6-3 主程序图

（2）控制器 S 函数子程序：chap6_1s. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
```

```
function [sys,x0,str,ts] = mdlInitializeSizes
global M V x0 fai

sizes = simsizes;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 1;
sizes.NumInputs       = 3;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 1;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [0 0];
function sys = mdlOutputs(t,x,u)
c1 = 35;
c2 = 15;

zd = u(1);
dzd = 6 * pi * cos(6 * pi * t);
ddzd = - (6 * pi)^2 * sin(6 * pi * t);
x1 = u(2);
x2 = u(3);

f = - 25 * x2;
b = 133;

z1 = x1 - zd;
dz1 = x2 - dzd;

alfa1 = - c1 * z1 + dzd;
z2 = x2 - alfa1;
ut = (1/b) * ( - f - c2 * z2 - z1 - c1 * dz1 + ddzd);

sys(1) = ut;
```

(3) 被控对象子程序:chap6_1plant. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
```

```
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [0.5;0];

str = [];
ts = [0 0];
function sys = mdlDerivatives(t,x,u)

sys(1) = x(2);
sys(2) = -25 * x(2) + 133 * u;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(4) 作图程序：chap6_1plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');
```

6.2 自适应反演滑模控制

在实际控制中，不确定性及外加干扰通常未知，因此，其中总不确定性 F 的上界很难确定。采用自适应方法可实现对 F 的估计[1]。

6.2.1 系统描述

被控对象为

$$\begin{cases}\dot{X}_U = X_P \\ \dot{X}_P = (A_m + \Delta A)X_P + (B_m + \Delta B)U_P + d(t) \\ Y = X_U\end{cases} \tag{6.13}$$

其中 $d(t)$ 为外加干扰。

将式(6.13)写为

$$\dot{X}_P = A_m X_P + B_m U_P + F \tag{6.14}$$

其中 F 为总不确定性，其表达式为

$$F = \Delta A X_P + \Delta B U_P + d(t) \tag{6.15}$$

其中 $|F| \leqslant \bar{F}$，ΔA 和 ΔB 为系统参数不确定部分。

假设参数不确定部分及外加干扰项变化缓慢，取

$$\dot{F} = 0 \tag{6.16}$$

6.2.2 Backstepping 滑模控制器的设计

假设位置指令为 Y_d，控制器设计步骤如下：

步骤一：

对于位置跟踪，跟踪误差为

$$z_1 = Y - Y_d \tag{6.17}$$

则

$$\dot{z}_1 = \dot{Y} - \dot{Y}_d = X_P - \dot{Y}_d \tag{6.18}$$

定义稳定项

$$\alpha_1 = c_1 z_1 \tag{6.19}$$

其中 c_1 为正的常数。

定义 Lyapunov 函数

$$V_1 = \frac{1}{2} z_1^2 \tag{6.20}$$

定义

$$z_2 = \dot{z}_1 + \alpha_1 = X_P - \dot{Y}_d + \alpha_1 \tag{6.21}$$

则

$$\dot{V}_1 = z_1(X_P - \dot{Y}_d) = z_1(z_2 - \alpha_1) = z_1 z_2 - c_1 z_1^2 \tag{6.22}$$

步骤二：

$$\dot{z}_2 = \dot{X}_P - \ddot{Y}_d + \dot{\alpha}_1 = A_m X_P + B_m U_P + F - \ddot{Y}_d + \dot{\alpha}_1 \tag{6.23}$$

定义 Lyapunov 函数

$$V_2 = V_1 + \frac{1}{2}\sigma^2 \tag{6.24}$$

其中 σ 为切换函数。定义切换函数为

$$\sigma = k_1 z_1 + z_2 \tag{6.25}$$

其中 $k_1 > 0$。则

$$\begin{aligned}\dot{V}_2 &= \dot{V}_1 + \sigma\dot{\sigma} = z_1 z_2 - c_1 z_1^2 + \sigma\dot{\sigma} \\ &= z_1 z_2 - c_1 z_1^2 + \sigma(k_1 \dot{z}_1 + \dot{z}_2)\end{aligned}$$

$$
\begin{aligned}
&= z_1 z_2 - c_1 z_1^2 + \sigma[k_1(z_2 - c_1 z_1) + A_m X_p + B_m U_p + F - \ddot{Y}_d + \dot{\alpha}_1] \\
&= z_1 z_2 - c_1 z_1^2 + \sigma[k_1(z_2 - c_1 z_1) + A_m(z_2 + \dot{Y}_d - \alpha_1) + B_m U_p + F - \ddot{Y}_d + \dot{\alpha}_1]
\end{aligned} \tag{6.26}
$$

设计控制器为

$$
U_p = B_m^{-1}[-k_1(z_2 - c_1 z_1) - A_m(z_2 + \dot{Y}_d - \dot{\alpha}_1) - \bar{F}\mathrm{sgn}(\sigma) + \ddot{Y}_d - \dot{\alpha}_1 - h(\sigma + \beta\mathrm{sgn}(\sigma))] \tag{6.27}
$$

其中 h 和 β 为正的常数。

由于控制器采用 $3F$ 的上界，当 F 为未知时，易造成抖振。

将式(6.27)代入式(6.26)得

$$
\begin{aligned}
\dot{V}_2 &= z_1 z_2 - c_1 z_1^2 - h\sigma^2 - h\beta|\sigma| + F\sigma - \bar{F}|\sigma| \\
&\leqslant -c_1 z_1^2 + z_1 z_2 - h\sigma^2 - h\beta|\sigma| + |\sigma|(|F| - \bar{F}) \\
&\leqslant -c_1 z_1^2 + z_1 z_2 - h\sigma^2 - h\beta|\sigma|
\end{aligned} \tag{6.28}
$$

取

$$
Q = \begin{bmatrix} c_1 + hk_1^2 & hk_1 - \frac{1}{2} \\ hk_1 - \frac{1}{2} & h \end{bmatrix} \tag{6.29}
$$

由于

$$
\begin{aligned}
z^T Q z &= [z_1 \quad z_2]\begin{bmatrix} c_1 + hk_1^2 & hk_1 - \frac{1}{2} \\ hk_1 - \frac{1}{2} & h \end{bmatrix}[z_1 \quad z_2]^T \\
&= c_1 z_1^2 + hk_1^2 z_1^2 + 2hk_1 z_1 z_2 - z_1 z_2 + hz_2^2 \\
&= c_1 z_1^2 - z_1 z_2 + h\sigma^2
\end{aligned} \tag{6.30}
$$

其中 $z^T = [z_1 \quad z_2]$。故式(6.28)可写为

$$
\dot{V}_2 \leqslant -z^T Q z - h\beta|\sigma|
$$

又由于

$$
|Q| = h(c_1 + hk_1^2) - \left(hk_1 - \frac{1}{2}\right)^2 = h(c_1 + k_1) - \frac{1}{4} \tag{6.31}
$$

通过取 h、c_1 和 k_1 的值，可使 $|Q|>0$，从而保证 Q 为正定矩阵。则

$$
\dot{V}_2 \leqslant 0 \tag{6.32}
$$

6.2.3 仿真实例

被控对象为

$$
\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -25x_1 + 133u + F(t) \end{cases} \tag{6.33}
$$

其中 $F(t)$ 为总的不确定性，$A_m = -25$，$B_m = 133$。

取 $F(t) = 2\sin(t)$。位置指令取 $r = \sin(2\pi t)$。采用控制律式(6.27)，取 $\bar{F} = 2.0$、$h = 20$、$c_1 = 70$、$k_1 = 30$、$\beta = 1.5$。仿真结果如图 6-4 和图 6-5 所示。

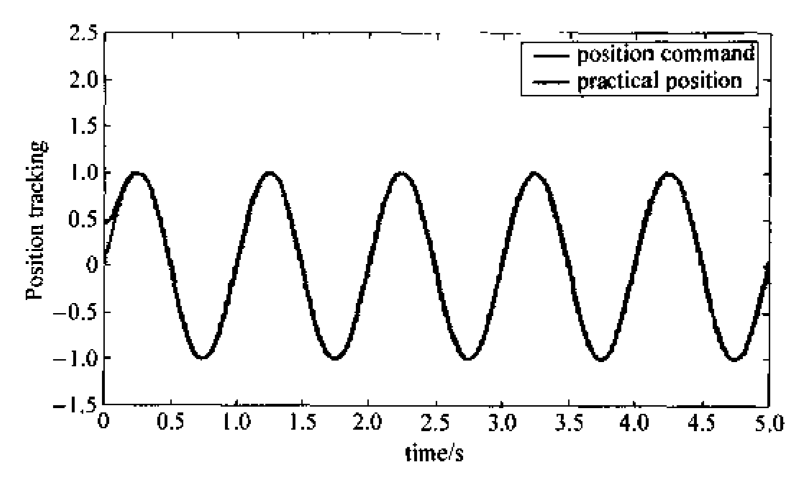


图6-4 位置跟踪

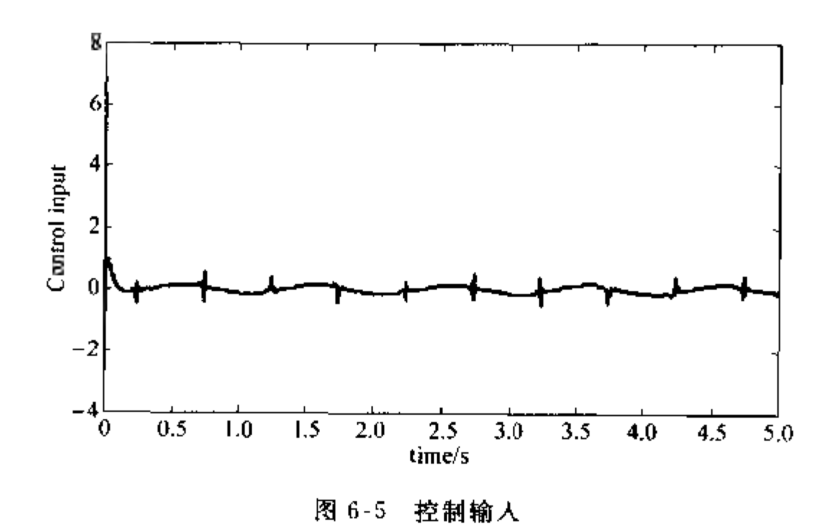


图6-5 控制输入

仿真程序：

(1) Simulink 主程序(如图 6-6 所示)：chap6_2sim. mdl

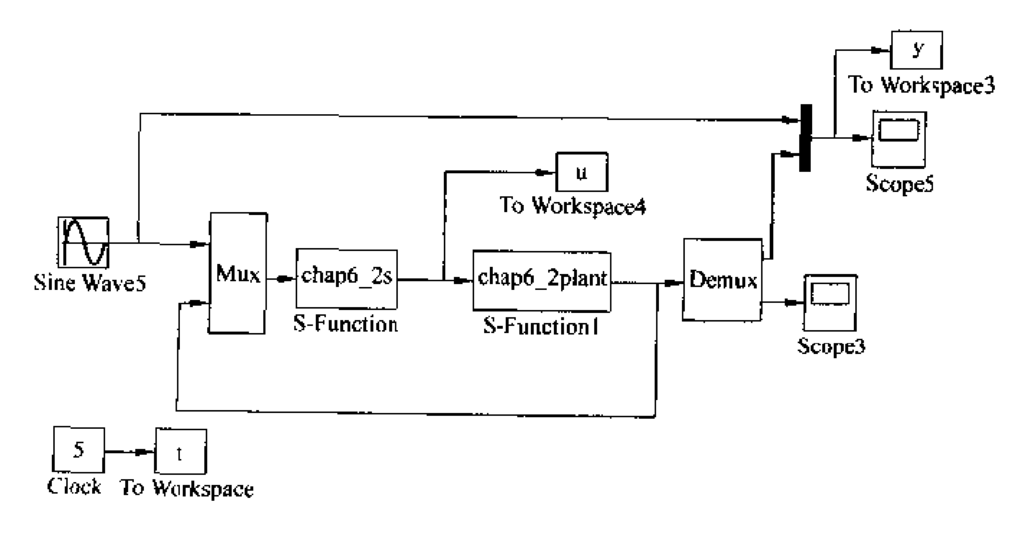


图6-6 主程序图

(2) 控制器 S 函数:chap6_2s. m

```
function [sys,x0,str,ts] = controller(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9},
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 3;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [];

function sys=mdlOutputs(t,x,u)
c1=70;
k1=50;
h=20;
beta=1.5;
Fmax=2;

Am=-25;
Bm=133;

Yd=u(1);
dYd=2*pi*cos(2*pi*t);
ddYd=-(2*pi)^2*sin(2*pi*t);

Y=u(2);
Xp=u(3);

z1=Y-Yd;
dz1=Xp-dYd;
alfa1=c1*z1;
d_alfa1=c1*dz1;

z2=Xp-dYd+alfa1;
```

```
rou = k1 * z1 + z2;

ut = 1/Bm * ( - k1 * (z2 - c1 * z1) - Am * (z2 + dYd - alfa1) + ...
    - Fmax * sign(rou) + ddYd - d_alfa1 - h * (rou + beta * sign(rou)));

sys(1) = ut;
```

(3) 被控对象 S 函数:chap6_2plant. m

```
% S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
  case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 },
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
  end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 2;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 1;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 0;

sys = simsizes(sizes);
x0 = [0.5,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
Am = -25;
Bm = 133;
F = 2 * sin(t);

sys(1) = x(2);
```

```
sys(2) = Am * x(1) + Bm * u + F;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(4) 作图程序：chap6_2plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');
```

6.2.4 自适应 Backstepping 滑模控制器的设计

为了避免采用 F 的上界，可采用自适应算法对 F 进行估计。

定义 Lyapunov 函数

$$V_3 = V_2 + \frac{1}{2\gamma}\tilde{F}^2 \tag{6.34}$$

其中 $\hat{F}$ 为 F 的估计值，F 的估计误差为 $\tilde{F} = F^* - \hat{F}$，$\gamma$ 为一个正的常数，V_2 式见(6.26)。

$$\begin{aligned}\dot{V}_3 &= \dot{V}_2 - \frac{1}{\gamma}\tilde{F}\dot{\hat{F}} \\ &= z_1 z_2 - c_1 z_1^2 + \sigma[k_1(z_2 - c_1 z_1) + A_m(z_2 + \dot{Y}_d - \alpha_1) + \\ &\quad B_m U_P + F - \ddot{Y}_d + \dot{\alpha}_1] - \frac{1}{\gamma}\tilde{F}\dot{\hat{F}} \\ &= z_1 z_2 - c_1 z_1^2 + \sigma[k_1(z_2 - c_1 z_1) + A_m(z_2 + \dot{Y}_d - \alpha_1) + \\ &\quad B_m U_P + \hat{F} - \ddot{Y}_d + \dot{\alpha}_1] - \frac{1}{\gamma}\tilde{F}(\dot{\hat{F}} - \gamma\sigma)\end{aligned} \tag{6.35}$$

设计自适应控制器为

$$U_P = B_m^{-1}[-k_1(z_2 - c_1 z_1) - A_m(z_2 + \dot{Y}_d - \alpha_1) - \hat{F} + \ddot{Y}_d - \dot{\alpha}_1 - h(\sigma + \beta\mathrm{sgn}(\sigma))] \tag{6.36}$$

设计自适应律为

$$\dot{\hat{F}} = \gamma\sigma \tag{6.37}$$

将式(6.36)和式(6.37)代入式(6.35)得

$$\dot{V}_3 = z_1 z_2 - c_1 z_1^2 - h\sigma^2 - h\beta|\sigma| \tag{6.38}$$

根据式(6.30)，式(6.38)可写为

$$\dot{V}_3 = -\boldsymbol{z}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{z} - h\beta|\sigma| \leqslant 0 \tag{6.39}$$

其中

$$Q=\begin{bmatrix} c_1+hk_1^2 & hk_1-\frac{1}{2} \\ hk_1-\frac{1}{2} & h \end{bmatrix} \tag{6.40}$$

通过取 h,c_1 和 k_1 的值,可使$|Q|>0$,从而保证 Q 为正定矩阵。

6.2.5　仿真实例

被控对象为

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=-25x_1+133u+F(t)\end{cases} \tag{6.41}$$

其中 $F(t)$为外部干扰。

取 $F(t)=2\sin(t)$。位置指令取 $r=A\sin(2\pi Ft)$,$A=1.0$,$F=1.0\text{Hz}$。采用控制律式(6.36)和自适应律式(6.37),取 $\gamma=30$,$h=20$,$c_1=70$,$k_1=50$。仿真结果如图 6-7～图 6-9 所示。

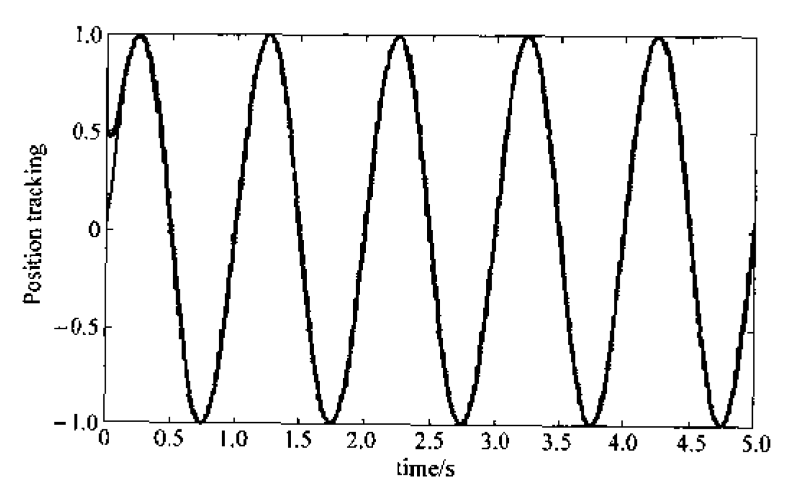


图 6-7　位置跟踪

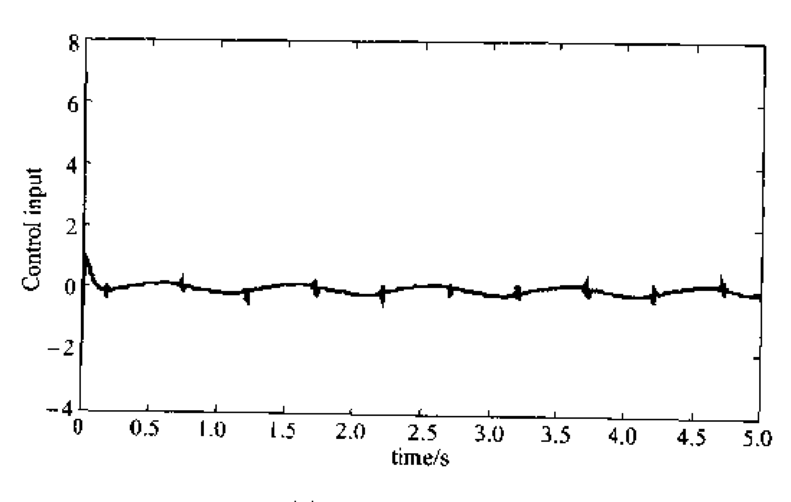


图 6-8　控制输入

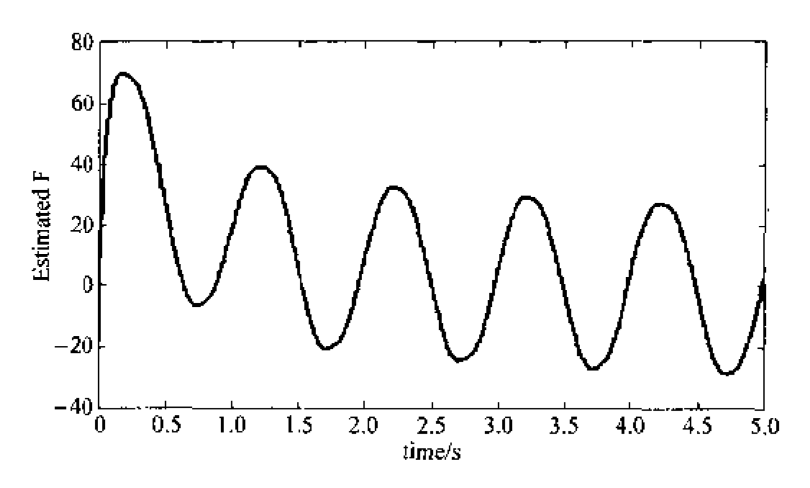


图 6-9　$\hat{F}$ 的自适应变化

仿真程序：

（1）Simulink 主程序（如图 6-10 所示）：chap6_3sim.mdl

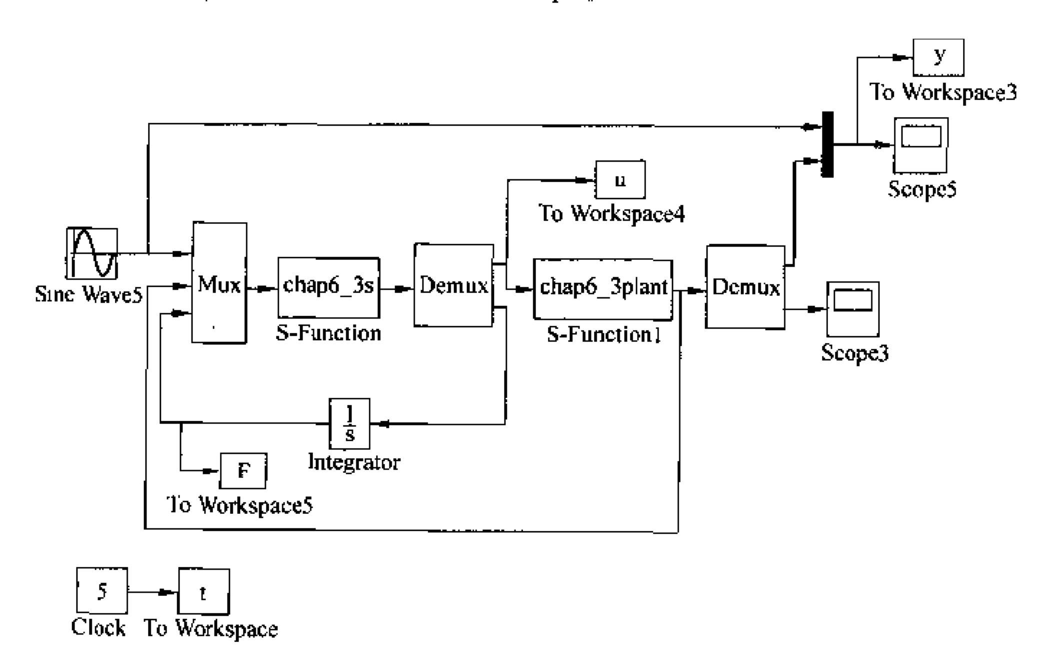


图 6-10　主程序图

（2）控制器 S 函数：chap6_3s.m

```
function [sys,x0,str,ts] = controller(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9},
    sys = [];
otherwise
```

```
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [];

function sys = mdlOutputs(t,x,u)
c1 = 70;
k1 = 50;
h = 20;
gama = 30;
beta = 1.5;
Fmax = 2;

Am = -25;
Bm = 133;

Yd = u(1);
dYd = 2*pi*cos(2*pi*t);
ddYd = -(2*pi)^2*sin(2*pi*t);

Y = u(2);
Xp = u(3);
Fp = u(4);

z1 = Y - Yd;
dz1 = Xp - dYd;
alfa1 = c1*z1;
d_alfa1 = c1*dz1;

z2 = Xp - dYd + alfa1;

rou = k1*z1 + z2;

ut = 1/Bm*(-k1*(z2 - c1*z1) - Am*(z2 + cYd - alfa1) + ...
    - Fp + ddYd - d_alfa1 - h*(rou + beta*sign(rou)));

sys(1) = ut;
sys(2) = gama*rou;
```

(3) 被控对象 S 函数：chap6_3plant.m

```
%S-function for continuous state equation
function [sys,x0,str,ts]=s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
  case 1,
    sys=mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys=mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 },
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
  end

%mdlInitializeSizes
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys=simsizes(sizes);
x0=[0.5,0];
str=[];
ts=[];

function sys=mdlDerivatives(t,x,u)
Am=-25;
Bm=133;
F=2*sin(t);

sys(1)=x(2);
sys(2)=Am*x(1)+Bm*u+F;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
```

(4) 作图程序:chap6_3plot.m

```
close all;

figure(1);
```

```
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,F(:,1),'r');
xlabel('time(s)');ylabel('Estimated F');
```

6.3 基于名义模型的反演滑模控制

将 backstepping 控制方法与滑模控制方法相结合，针对名义模型设计 backstepping 控制器，实际系统的不确定部分通过滑模控制器来补偿，从而可实现不确定系统的鲁棒控制。

6.3.1 系统描述

假设一个实际的伺服系统为

$$J\ddot{\theta} + B\dot{\theta} = u + d \tag{6.42}$$

其中 J 为转动惯量，B 为阻尼系数，u 为控制输入，d 为外加干扰，θ 为转动角度。$J>0$，$B>0$。

在实际伺服系统中，转动惯量 J 为时变的，并存在外加干扰、不确定性和未建模特性。图 6-11 所示为一个实际伺服系统的 Bode 图，采用 Bode 图逼近方法，可得到对象的名义模型：

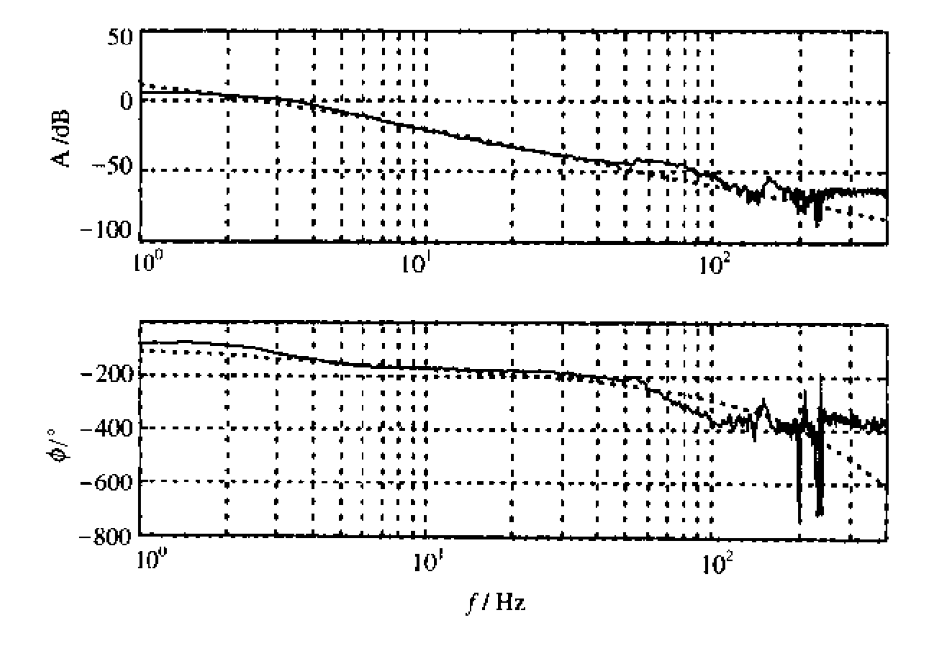


图 6-11　一个实际伺服系统 Bode 图

$$J_n\ddot{\theta}_n + B_n\dot{\theta}_n = \mu \tag{6.43}$$

其中 J_n 为名义模型转动惯量，B_n 为名义模型阻尼系数，μ 为名义模型控制输入，θ_n 为转动角度。$J_n>0$，$B_n>0$。

6.3.2 控制系统结构

控制系统结构如图 6-12 所示。该系统由两个控制器构成：采用滑模控制器控制实际对象，实现 $\theta\to\theta_n$；采用 backstepping 控制器控制名义模型，实现 $\theta_n\to\theta_r$。从而达到 $\theta\to\theta_r$ 的目的。

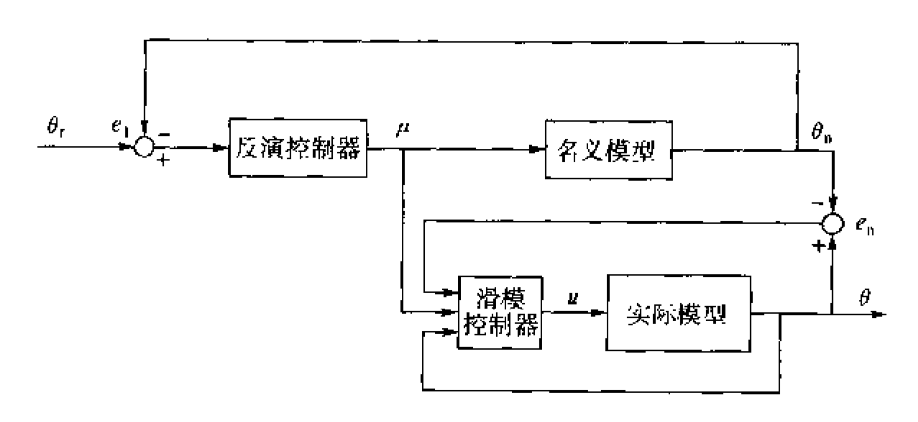


图 6-12 控制系统结构

6.3.3 名义模型 backstepping 控制器的设计

参考文献[2]，采用积分 backstepping 控制方法设计名义模型的控制器。

定义位置误差 $e_1=\theta_r-\theta_n$，则

$$\frac{\mathrm{d}e_1}{\mathrm{d}t} = \dot{\theta}_r - \dot{\theta}_n = \omega_r - \omega_n \tag{6.44}$$

其中 $\omega_r=\dot{\theta}_r$，$\omega_n=\dot{\theta}_n$。

按传统 backstepping 设计方法，如果取 $\omega_n=c_1e_1+\dot{\theta}_r$，$c_1$ 为正的常数，则式(6.44)变为 $\frac{\mathrm{d}e_1}{\mathrm{d}t}=-c_1e_1$。名义模型速度 ω_n 无法人为定义，但理想的速度 ω_r 可以定义。定义理想速度 ω_r 为

$$\omega_r = c_1e_1 + \dot{\theta}_r + \lambda_1 z_1 \tag{6.45}$$

其中 λ_1 为正的常数，z_1 为积分项。

积分项 z_1 定义为

$$z_1 = \int_0^t e_1(t)\mathrm{d}t \tag{6.46}$$

名义模型速度 ω_n 与理想速度 ω_r 之间的误差为

$$e_2 = \omega_r - \omega_n = c_1e_1 + \dot{\theta}_r + \lambda_1 z_1 - \omega_n = c_1e_1 + \lambda_1 z_1 + \frac{\mathrm{d}e_1}{\mathrm{d}t} \tag{6.47}$$

则

$$\frac{\mathrm{d}e_2}{\mathrm{d}t} = c_1 \frac{\mathrm{d}e_1}{\mathrm{d}t} + \ddot{\theta}_r + \lambda_1 e_1 - \frac{\mu}{J_n} + \frac{B_n}{J_n}\dot{\theta}_n \tag{6.48}$$

由式(6.47)得

$$\frac{\mathrm{d}e_1}{\mathrm{d}t} = -c_1 e_1 - \lambda_1 z_1 + e_2 \tag{6.49}$$

将式(6.49)代入式(6.48)得

$$\frac{\mathrm{d}e_2}{\mathrm{d}t} = c_1(-c_1 e_1 - \lambda_1 z_1 + e_2) + \ddot{\theta}_r + \lambda_1 e_1 - \frac{\mu}{J_n} + \frac{B_n}{J_n}\dot{\theta}_n \tag{6.50}$$

定义 Lyapunov 函数为

$$V_1 = \frac{\lambda_1}{2}z_1^2 + \frac{1}{2}e_1^2 + \frac{1}{2}e_2^2 \tag{6.51}$$

则

$$\begin{aligned}
\dot{V}_1 &= \lambda_1 z_1 \dot{z}_1 + e_1 \frac{\mathrm{d}e_1}{\mathrm{d}t} + e_2 \frac{\mathrm{d}e_2}{\mathrm{d}t} \\
&= \lambda_1 z_1 e_1 + e_1(-c_1 e_1 - \lambda_1 z_1 + e_2) \\
&\quad + e_2\left[c_1(-c_1 e_1 - \lambda_1 z_1 + e_2) + \ddot{\theta}_r + \lambda_1 e_1 - \frac{\mu}{J_n} + \frac{B_n}{J_n}\dot{\theta}_n\right] \\
&= e_1(-c_1 e_1 + e_2) + e_2\left[c_1(-c_1 e_1 - \lambda_1 z_1 + e_2) + \ddot{\theta}_r + \lambda_1 e_1 - \frac{\mu}{J_n} + \frac{B_n}{J_n}\dot{\theta}_n\right]
\end{aligned} \tag{6.52}$$

设计 backstepping 控制律为

$$\mu = J_n[(1 - c_1^2 + \lambda_1)e_1 + (c_1 + c_2)e_2 - c_1\lambda_1 z_1 + \ddot{\theta}_r] + B_n\dot{\theta}_n \tag{6.53}$$

其中 c_2 为正的常数。

将式(6.53) 代入 $\dot{V}_1$ 得

$$\dot{V}_1 = -c_1 e_1^2 - c_2 e_2^2 \leqslant 0 \tag{6.54}$$

6.3.4 实际对象全局滑模控制器的设计

假设被控对象参数满足

$$J_m \leqslant J \leqslant J_M \tag{6.55}$$

$$B_m \leqslant B \leqslant B_M \tag{6.56}$$

$$|d| \leqslant d_M \tag{6.57}$$

设名义模型与实际对象之间的误差为

$$e_n = \theta - \theta_n \tag{6.58}$$

定义全局滑模函数为

$$s = \dot{e}_n + \lambda e_n - f(t) \tag{6.59}$$

其中 $\lambda > 0$。

为了实现全局鲁棒特性，要求函数 $f(t)$ 满足三个条件：(a) $f(0) = \dot{e}_{n0} + \lambda e_{n0}$；(b) 当 $t \to \infty$ 时 $f(t) \to 0$；(c) $f(t)$ 存在对时间的一阶导数。其中 e_{n0} 和 $\dot{e}_{n0}$ 为 $t=0$ 时的误差及误差变化率。条件(a)保证了系统状态初始位置位于滑模面上，条件(b)保证了闭环系统的稳定性，条件(c)表明了滑模存在的条件。

定义函数 $f(t)$ 为

$$f(t)=s(0)\exp(-\eta t) \tag{6.60}$$

其中 $\eta>0$，$s(0)$ 为 s 的初值。

定义 λ 为

$$\lambda=\frac{B_n}{J_n} \tag{6.61}$$

定义中间值

$$J_a=\frac{1}{2}(J_m+J_M) \tag{6.62}$$

$$B_a=\frac{1}{2}(B_m+B_M) \tag{6.63}$$

设计滑模控制律为

$$u=-Ks-h(\theta,\dot{\theta},t)\cdot\operatorname{sgn}(s)+J_a\left(\frac{1}{J_n}\mu-\lambda\dot{\theta}\right)+B_a\dot{\theta} \tag{6.64}$$

其中 $K>0$。将 $h(\theta,\dot{\theta},t)$ 定义为

$$h(\theta,\dot{\theta},t)=d_M+\frac{1}{2}(J_M-J_m)\left|\frac{1}{J_n}\mu-\lambda\dot{\theta}\right|+\frac{1}{2}(B_M-B_m)\,|\dot{\theta}|+J_M\eta\,|s(0)|\exp(-\eta t) \tag{6.65}$$

稳定性分析：

定义 Lyapunov 函数为

$$V_2=\frac{1}{2}Js^2 \tag{6.66}$$

由式(6.59)，得

$$\begin{aligned}J\dot{s}&=J[(\ddot{\theta}-\ddot{\theta}_n)+\lambda(\dot{\theta}-\dot{\theta}_n)+\eta s(0)\exp(-\eta t)]\\&=(J\ddot{\theta}+B\dot{\theta})-\frac{J}{J_n}(J_n\ddot{\theta}_n+B_n\dot{\theta}_n)-B\dot{\theta}+\lambda J\dot{\theta}+J\eta s(0)\exp(-\eta t)\\&=u+d-\frac{J}{J_n}\mu-B\dot{\theta}+\lambda J\dot{\theta}+J\eta s(0)\exp(-\eta t)\end{aligned} \tag{6.67}$$

将式(6.64)代入式(6.67)得

$$\begin{aligned}J\dot{s}&=-Ks-h(\theta,\dot{\theta},t)\operatorname{sgn}(s)+J_a\left(\frac{1}{J_n}\mu-\lambda\dot{\theta}\right)+B_a\dot{\theta}+d-B\dot{\theta}-\frac{J}{J_n}\mu+\lambda J\dot{\theta}+J\eta s(0)\exp(-\eta t)\\&=-Ks-h(\theta,\dot{\theta},t)\operatorname{sgn}(s)+d+(J_a-J)\left(\frac{1}{J_n}\mu-\lambda\dot{\theta}\right)+(B_a-B)\dot{\theta}+J\eta s(0)\exp(-\eta t)\end{aligned} \tag{6.68}$$

则

$$\begin{aligned}\dot{V}_2=Js\dot{s}&=-Ks^2-h(\theta,\dot{\theta},t)\,|s|+s\Big[d+(J_a-J)\left(\frac{1}{J_n}\mu-\lambda\dot{\theta}\right)+\\&\qquad(B_a-B)\dot{\theta}+J\eta s(0)\exp(-\eta t)\Big]\\&\leqslant-Ks^2-h(\theta,\dot{\theta},t)\,|s|+|s|\Big[|d|+|J_a-J|\left|\frac{1}{J_n}\mu-\lambda\dot{\theta}\right|+\\&\qquad|B_a-B|\,|\dot{\theta}|+J\eta\,|s(0)|\exp(-\eta t)\Big]\end{aligned} \tag{6.69}$$

根据式(6.62)和式(6.63)，有

$$|J_a - J| \leqslant \frac{1}{2}(J_M - J_m) \tag{6.70}$$

$$|B_a - B| \leqslant \frac{1}{2}(B_M - B_m) \tag{6.71}$$

因此有

$$h(\theta,\dot{\theta},t) \geqslant |d| + |J_a - J| \left|\frac{1}{J_n}\mu - \lambda\dot{\theta}\right| + |B_a - B||\dot{\theta}| + J\eta|s(0)|\exp(-\eta t) \tag{6.72}$$

则

$$\dot{V}_2 \leqslant -Ks^2 \tag{6.73}$$

由式(6.66)和式(6.73)得

$$s(t) \leqslant |s(0)|\exp\left(-\frac{K}{J}t\right) \tag{6.74}$$

式(6.74)表明，$s(t)$以指数的形式趋近于零。

6.3.5　仿真实例

实际被控对象为

$$J\ddot{\theta} + B\dot{\theta} = u + d(t) \tag{6.75}$$

设 $B=\frac{25+10\sin(2\pi t)}{133}$，$J=\frac{1}{133+30\sin(2\pi t)}$，$d(t)$为最大值为1的随机干扰。

在控制律式(6.53)中，$J_n=\frac{1}{133}$，$B_n=\frac{25}{133}$，控制器参数取 $c_1=100$，$c_2=100$，$\lambda_1=5$。

在控制律式(6.64)中，$J_m=\frac{1}{163}$，$J_M=\frac{1}{103}$，$B_m=\frac{15}{163}$，$B_M=\frac{35}{103}$，$d_M=1.0$，$\lambda=\frac{B_n}{J_n}=25$。取$K=20$，$\eta=10$。

位置指令为 $\theta_r(t)=A\sin(2\pi Ft)$，$A=0.5$，$F=3.0\text{Hz}$。系统初始状态为[0.5　0]。仿真结果如图6-13～图6-16所示。

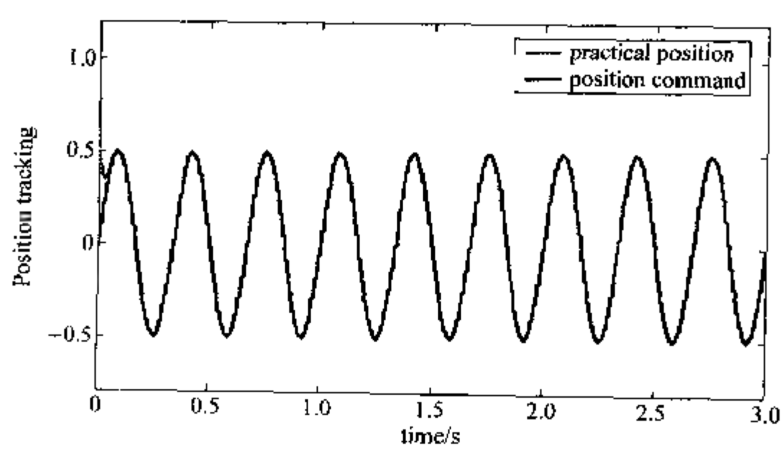


图6-13　正弦位置跟踪

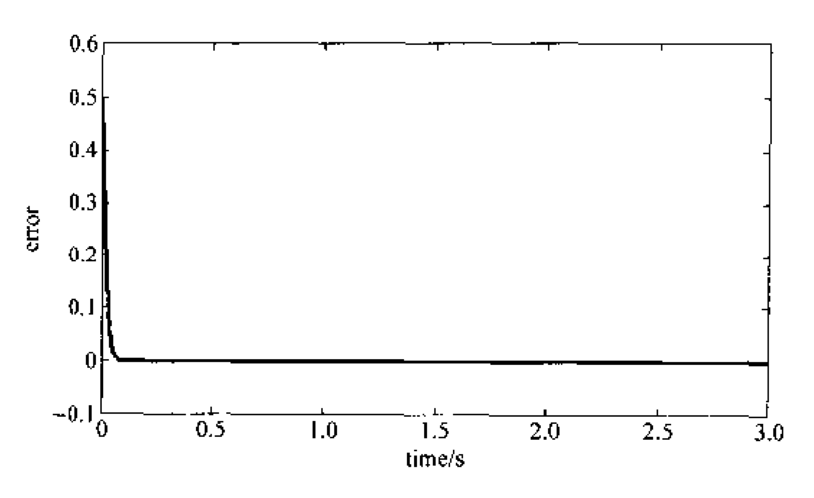


图 6-14 位置跟踪误差

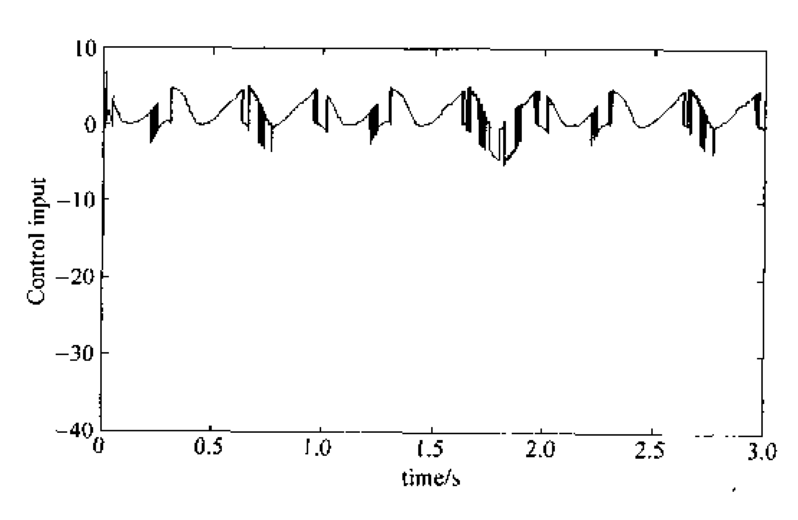


图 6-15 控制输入信号

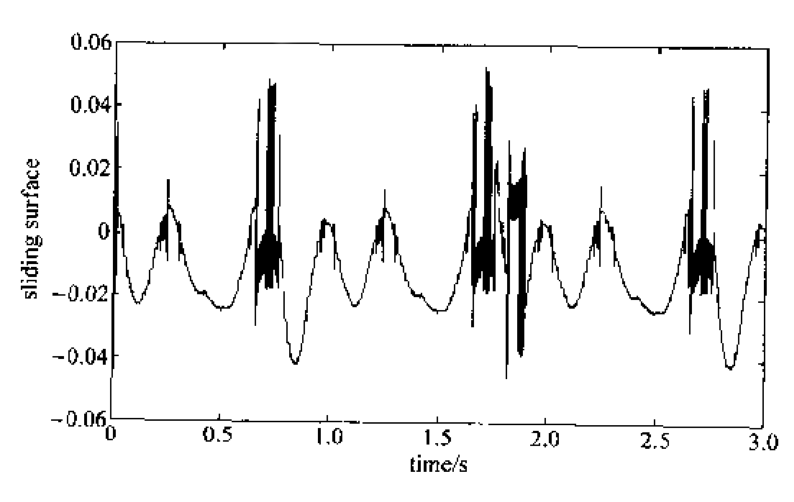


图 6-16 滑模切换函数

仿真程序：

(1) Simulink 主程序(如图 6-17 所示)：chap6_4sim. mdl

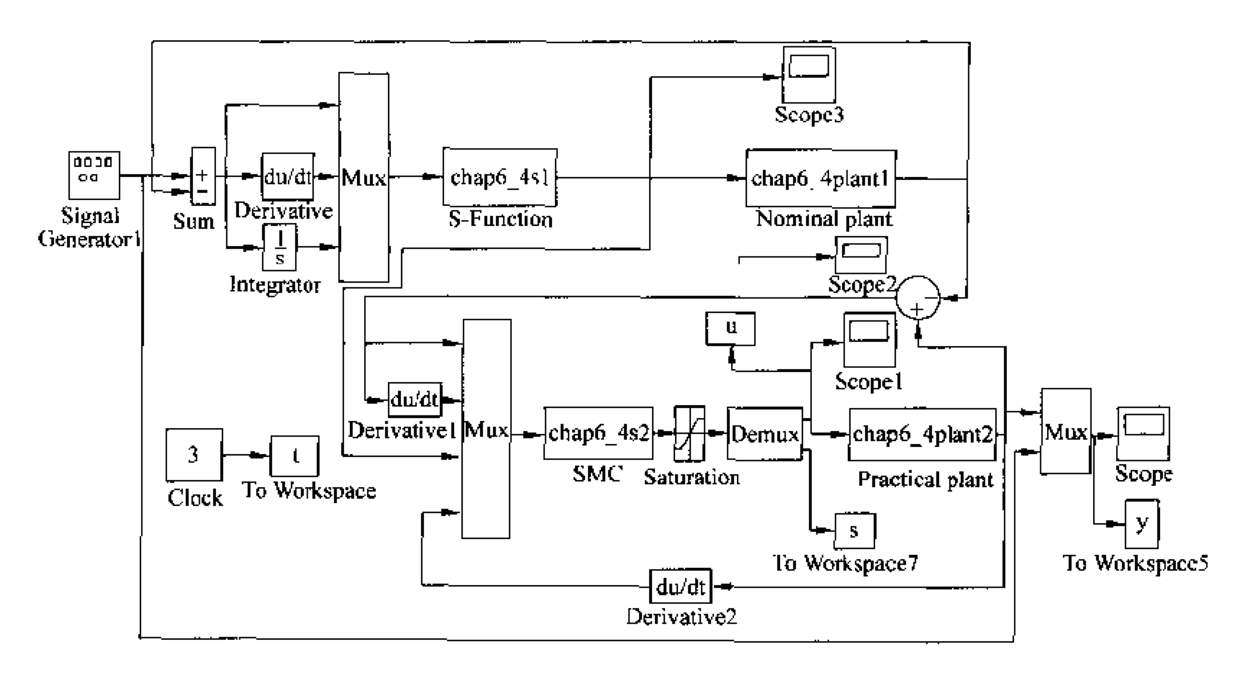


图 6-17　主程序图

(2) Backstepping 控制器 S 函数:chap6_4s1. m

```
%S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 0;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 1;
sizes.NumInputs       = 3;
sizes.DirFeedthrough  = 1;
sizes.NumSampleTimes  = 0;
```

```
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)
error = u(1);
derror = u(2);

c1 = 100;c2 = 100;n1 = 5;

r = 0.5 * sin(3 * 2 * pi * t);
dr = 0.5 * 3 * 2 * pi * cos(3 * 2 * pi * t);
ddr = - 0.5 * (3 * 2 * pi)^2 * sin(3 * 2 * pi * t);

e1 = u(1);
de = u(2);
z1 = u(3);

x2 = dr - de;
w = x2;

wr = c1 * e1 + dr + n1 * z1;

e2 = wr - w;

J = 1/133;
b = 25/133;
ut = J * [(1 - c1^2 + n1) * e1 + (c1 + c2) * e2 - c1 * n1 * z1 + ddr] + b * x2;

sys(1) = ut;
```

(3) 名义模型 *S* 函数:chap6_4plant1.m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
```

```
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0.5,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)

sys(1) = x(2);
sys(2) = -25 * x(2) + 133 * u;

function sys = mdlOutputs(t,x,u)

sys(1) = x(1);
```

(4) 滑模控制器 S 函数：chap6_4s2.m

```
%S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
```

```
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)
persistent s0

Bn = 25/133;
Jn = 1/133;
lamt = Bn/Jn;

Jm = 1/163;JM = 1/103;
Bm = 15/163;BM = 35/103;

dM = 1.0;
K = 20;

e = u(1);
de = u(2);
nu = u(3);
v = u(4);

s1 = de + lamt * e;

if t == 0
   e0 = e;de0 = de;
     s0 = de0 + lamt * e0;
end

temp0 = (1/Jn) * nu-lamt * v;

Ja = 1/2 * (JM + Jm);
Ba = 1/2 * (BM + Bm);

xite = 10;
ft = s0 * exp( - xite * t);
s = s1-ft;

H = dM + 1/2 * (JM-Jm) * abs(temp0) + 1/2 * (BM-Bm) * abs(v) + JM * xite * abs(s0) * exp( - xite * t);

ut = - K * s - H * sign(s) + Ja * ((1/Jn) * nu-lamt * v) + Ba * v;

sys(1) = ut;
sys(2) = s;
```

(5) 实际模型 S 函数:chap6_4plant2. m

```
%S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0.5,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
sys(1) = x(2);
sys(2) = -(25 + 10 * sin(2 * pi * t)) * x(2) + (133 + 30 * sin(2 * pi * t)) * u + 1.0 * rands(1);

function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
```

(6) 作图程序:chap6_4plot. m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('position tracking');
```

```
figure(2);
plot(t,y(:,1) - y(:,2),'r');
xlabel('time(s)');ylabel('error');

figure(3);
plot(t,u,'r');
xlabel('time(s)');ylabel('control input');

figure(4);
plot(t,s,'r');
xlabel('time(s)');ylabel('sliding surface');
```

6.4 移动机器人的滑模轨迹跟踪控制

移动机器人可通过移动来完成一些比较危险的任务，如地雷探测、海底探测、无人机驾驶等，在工业、国防等很多领域都有实用价值。移动机器人有多种，最常见的是在地面上依靠轮子移动的机器人，也称作“无人驾驶车”或“移动小车”。

6.4.1 移动机器人运动学模型

图 6-18 所示的移动机器人为轮式移动机器人，该机器人两个较大的后轮为驱动轮，两个较小的后轮为从动轮。左右两个后轮各由一个电机来驱动，如果两个电机的转速不同，则左右两个后轮会产生“差动”，从而可实现转弯。

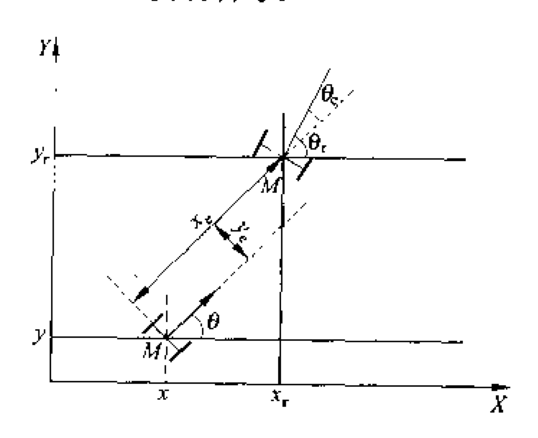


图 6-18　移动机器人位姿误差坐标

移动机器人的状态由其两个驱动轮的轴中点 M 在坐标系的位置及航向 θ 来表示，令 $\boldsymbol{P}=(x \quad y \quad \theta)^{\mathrm{T}}$，$\boldsymbol{q}=(v \quad \omega)^{\mathrm{T}}$，其中 (x,y) 为移动机器人的位置，θ 为移动机器人前进方向与 X 轴的夹角，v 和 ω 分别为移动机器人的线速度和角速度，在运动学模型中它们是控制输入。

移动机器人的运动学方程为

$$\dot{\boldsymbol{p}}=\begin{bmatrix}\dot{x}\\\dot{y}\\\dot{\theta}\end{bmatrix}=\begin{bmatrix}\cos\theta & 0\\\sin\theta & 0\\0 & 1\end{bmatrix}\boldsymbol{q} \tag{6.76}$$

考虑对具有位姿指令 $\boldsymbol{p}_r=(x_r \quad y_r \quad \theta_r)^T$ 和速度指令 $\boldsymbol{q}_r=(v_r \quad \omega_r)^T$ 参考小车的跟踪。

在图 6-18 中，移动机器人从位姿 $\boldsymbol{p}=(x \quad y \quad \theta)^T$ 移动到位姿 $\boldsymbol{p}_r=(x_r \quad y_r \quad \theta_r)^T$，移动机器人在新坐标系 X_e-Y_e 中的坐标为

$$\boldsymbol{p}_e=(x_e \quad y_e \quad \theta_e)^T \tag{6.77}$$

其中 $\theta_e=\theta_r-\theta$。

设新坐标系 X_e-Y_e 与坐标系 $X-Y$ 之间的夹角为 θ。根据坐标变换公式，可得描述移动机器人位姿的误差方程为：

$$\boldsymbol{p}_e=\begin{bmatrix}x_e\\y_e\\\theta_e\end{bmatrix}=\begin{bmatrix}\cos\theta & \sin\theta & 0\\-\sin\theta & \cos\theta & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_r-x\\y_r-y\\\theta_r-\theta\end{bmatrix} \tag{6.78}$$

根据文献[3]，得位姿误差微分方程为

$$\dot{\boldsymbol{p}}_e=\begin{bmatrix}\dot{x}_e\\\dot{y}_e\\\dot{\theta}_e\end{bmatrix}=\begin{bmatrix}y_e\omega-v+v_r\cos\theta_e\\-x_e\omega+v_r\sin\theta_e\\\omega_r-\omega\end{bmatrix} \tag{6.79}$$

位姿误差微分方程(6.79)的推导过程如下：

由式(6.76)得

$$\dot{x}=v\cos\theta,\dot{y}=v\sin\theta \tag{6.80}$$

则

$$\dot{x}\cos\theta+\dot{y}\sin\theta=v \tag{6.81}$$

$$\dot{x}\sin\theta-\dot{y}\cos\theta=0 \tag{6.82}$$

由式(6.78)得

$$x_e=(x_r-x)\cos\theta+(y_r-y)\sin\theta$$

$$y_e=-(x_r-x)\sin\theta+(y_r-y)\cos\theta$$

$$\begin{aligned}\dot{x}_e&=(\dot{x}_r-\dot{x})\cos\theta-(x_r-x)\sin\theta\dot{\theta}+(\dot{y}_r-\dot{y})\sin\theta+(y_r-y)\cos\theta\dot{\theta}\\&=y_e\omega-(\dot{x}\cos\theta+\dot{y}\sin\theta)+(\dot{x}_r\cos\theta+\dot{y}_r\sin\theta)\\&=y_e\omega-v+\dot{x}_r\cos(\theta_r-\theta_e)+\dot{y}_r\sin(\theta_r-\theta_e)\\&=y_e\omega-v+(\dot{x}_r\cos\theta_r+\dot{y}_r\sin\theta_r)\cos\theta_e+(\dot{x}_r\sin\theta_r-\dot{y}_r\cos\theta_r)\sin\theta_e\\&=y_e\omega-v+v_r\cos\theta_e\end{aligned}$$

$$\begin{aligned}\dot{y}_e&=-(\dot{x}_r-\dot{x})\sin\theta-(x_r-x)\cos\theta\dot{\theta}+(\dot{y}_r-\dot{y})\cos\theta-(y_r-y)\sin\theta\dot{\theta}\\&=-\dot{x}_r\sin\theta+\dot{y}_r\cos\theta+(\dot{x}\sin\theta-\dot{y}\cos\theta)-(x_r\cos(\theta)+y_r\sin\theta)\dot{\theta}+(x\cos\theta+y\sin\theta)\dot{\theta}\\&=-\dot{x}_r\sin\theta+\dot{y}_r\cos\theta-[(x_r-x)\cos\theta+(y_r-y)\sin\theta]\dot{\theta}\\&=-\dot{x}_r\sin(\theta_r-\theta_e)+\dot{y}_r\cos(\theta_r-\theta_e)-x_e\omega\\&=-\dot{x}_r(\sin\theta_r\cos\theta_e-\cos\theta_r\sin\theta_e)+\dot{y}_r(\cos\theta_r\cos\theta_e+\sin\theta_r\sin\theta_e)-x_e\omega\\&=[-\dot{x}_r\sin\theta_r+\dot{y}_r\cos\theta_r]\cos\theta_e+[\dot{x}_r\cos\theta_r+\dot{y}_r\sin\theta_r]\sin\theta_e-x_e\omega\\&=0+v_r\sin\theta_e-x_e\omega=-x_e\omega+v_r\sin\theta_e\end{aligned}$$

移动机器人运动学模型的轨迹跟踪即寻找控制输入 $q=(v \quad \omega)^{\mathrm{T}}$，使对任意的初始误差，系统式(6.79)在该控制输入作用下，$p_e=(x_e \quad y_e \quad \theta_e)^{\mathrm{T}}$ 有界，且 $\lim\limits_{t\to\infty}\|(x_e \quad y_e \quad \theta_e)^{\mathrm{T}}\|=0$。

移动机器人运动学模型式(6.79)是一个多输入非线性系统，其切换函数的设计是一个难点。根据文献[4]，可采用 Backstepping 方法设计切换函数。

6.4.2 切换函数的设计

引理 1[5] 对任意 $x\in\mathbf{R}$ 且 $|x|<\infty$，有 $\phi(x)=x\sin(\arctan x)\geqslant 0$，当且仅当 $x=0$ 时“=”成立。

证明 分以下三种情况讨论：

(1) 当 $x=0$ 时，$\phi(0)=0$；

(2) 当 $x\in(0,+\infty)$ 时，有 $\arctan x\in(0,\pi/2)$，则 $\sin(\arctan x)>0$，即 $\phi(x)>0$；

(3) 当 $x\in(-\infty,0)$ 时，有 $\arctan x\in(-\pi/2,0)$，则 $\sin(\arctan x)<0$，即 $\phi(x)>0$。

根据引理 1，可设计基于 Backstepping 的滑模切换函数。其设计过程如下：

当 $x_e=0$ 时，考察 Lyapunov 函数

$$V_y=\frac{1}{2}y_e^2 \tag{6.83}$$

假设 $\theta_e=-\arctan(v_r y_e)$，则

$$\begin{aligned}\dot{V}_y&=y_e\dot{y}_e=y_e(-x_e w+v_r\sin\theta_e)=-y_e x_e w+v_r y_e\sin(-\arctan(v_r y_e))\\&=-y_e x_e w-v_r y_e\sin(\arctan(v_r y_e))\end{aligned}$$

由引理 1 可知

$$v_r y_e\sin(-\arctan(v_r y_e))\geqslant 0 \quad (\text{当且仅当 } v_r y_e=0 \text{ 时“=”式成立})$$

则

$$\dot{V}_y\leqslant 0$$

可得到结论：只要 x_e 收敛到零且 θ_e 收敛到 $-\arctan(v_r y_e)$，则系统状态 y_e 收敛到零。

根据该结论，可设计切换函数为

$$s=\begin{bmatrix}s_1\\s_2\end{bmatrix}=\begin{bmatrix}x_e\\\theta_e+\arctan(v_r y_e)\end{bmatrix} \tag{6.84}$$

通过设计滑模控制器，使 $s_1\to 0$，$s_2\to 0$，即实现 x_e 收敛到零且 θ_e 收敛到 $-\arctan(v_r y_e)$，从而实现 $y_e\to 0$ 和 $\theta_e\to 0$。

6.4.3 滑模控制器设计

取等速趋近律，令

$$\dot{s}=-k\,\mathrm{sgn}\,s \tag{6.85}$$

为了减弱抖振，采用连续函数取代符号函数

$$\dot{s}_i=-k_i\frac{s_i}{|s_i|-\delta_i} \quad i=1,2 \tag{6.86}$$

其中 δ_i 为正小数。

令 $\alpha=\arctan(v_r y_e)$，由式(6.79)和式(6.84)得

$$\dot{s}=\begin{bmatrix}\dot{s}_1\\ \dot{s}_2\end{bmatrix}=\begin{bmatrix}-k_1\dfrac{s_1}{|s_1|+\delta_1}\\ -k_2\dfrac{s_2}{|s_2|+\delta_2}\end{bmatrix}=\begin{bmatrix}\dot{x}_e\\ \dot{\theta}_e+\dfrac{\partial\alpha}{\partial v_r}\dot{v}_r+\dfrac{\partial\alpha}{\partial y_e}\dot{y}_e\end{bmatrix}$$

$$=\begin{bmatrix}y_e\omega-v+v_r\cos\theta_e\\ \omega_r-\omega+\dfrac{\partial\alpha}{\partial v_r}\dot{v}_r+\dfrac{\partial\alpha}{\partial y_e}(-x_e\omega+v_r\sin\theta_e)\end{bmatrix}\tag{6.87}$$

经整理，得控制律为

$$q=\begin{bmatrix}v\\ \omega\end{bmatrix}=\begin{bmatrix}y_e\omega+v_r\cos\theta_e+k_1\dfrac{s_1}{|s_1|+\delta_1}\\ \dfrac{\omega_r+\dfrac{\partial\alpha}{\partial v_r}\dot{v}_r+\dfrac{\partial\alpha}{\partial y_e}(v_r\sin\theta_e)+k_2\dfrac{s_2}{|s_2|+\delta_2}}{1+\dfrac{\partial\alpha}{\partial y_e}x_e}\end{bmatrix}\tag{6.88}$$

其中，$\dfrac{\partial\alpha}{\partial v_r}=\dfrac{y_e}{1+(v_r y_e)^2}$，$\dfrac{\partial\alpha}{\partial y_e}=\dfrac{v_r}{1+(v_r y_e)^2}$。

6.4.4　仿真实例

被控对象为位姿误差微分方程(6.79)，即

$$\begin{cases}\dot{x}_e=y_e\omega-v+v_r\cos\theta_e\\ \dot{y}_e=-x_e\omega+v_r\sin\theta_e\\ \dot{\theta}_e=\omega_r-\omega\end{cases}$$

采用如下两个实例进行仿真。

实例 1：

跟踪线速度和角速度均为匀速运动的圆轨迹。在主程序图中被控对象采用 S 函数 chap6_5plant.m 来表示。取 $w_r=1.0$，$v_r=1.0$，则 $\dot{v}_r=0$，半径为 $r=\dfrac{v_r}{\omega_r}=1.0$，位姿指令 $\boldsymbol{p}_r=(x_r\quad y_r\quad \theta_r)^{\mathrm{T}}$ 为

$$x_r=r\cos(\omega_r t)=\cos t$$
$$y_r=r\sin(\omega_r t)=\sin t$$
$$\theta_r=\omega_r t=t$$

取 $\delta_1=\delta_2=0.02$，$k_1=k_2=6.0$，位姿误差初始值为[3　0　0]，采用控制律式(6.88)，仿真结果如图 6-19～图 6-25 所示。

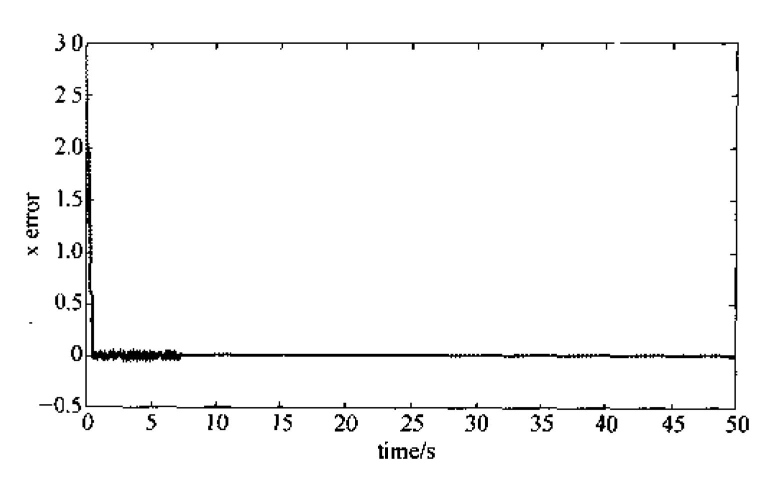


图 6-19 X 轴方向误差

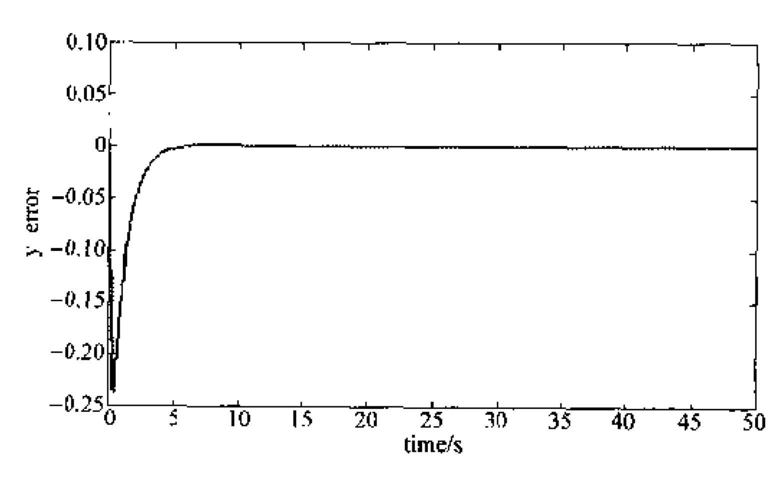


图 6-20 Y 轴方向误差

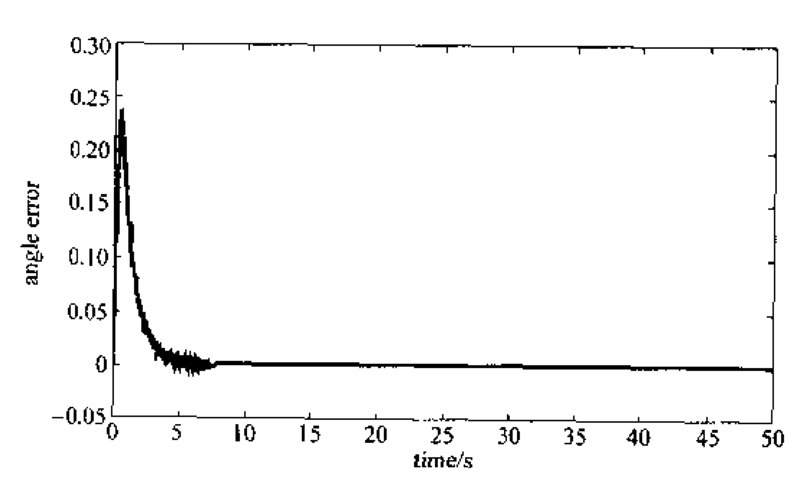


图 6-21 航向角方向误差

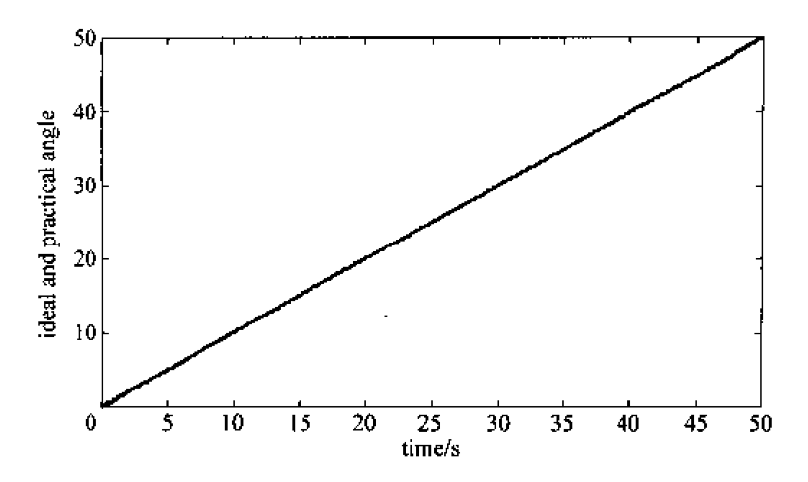


图 6-22　航向角的位置跟踪

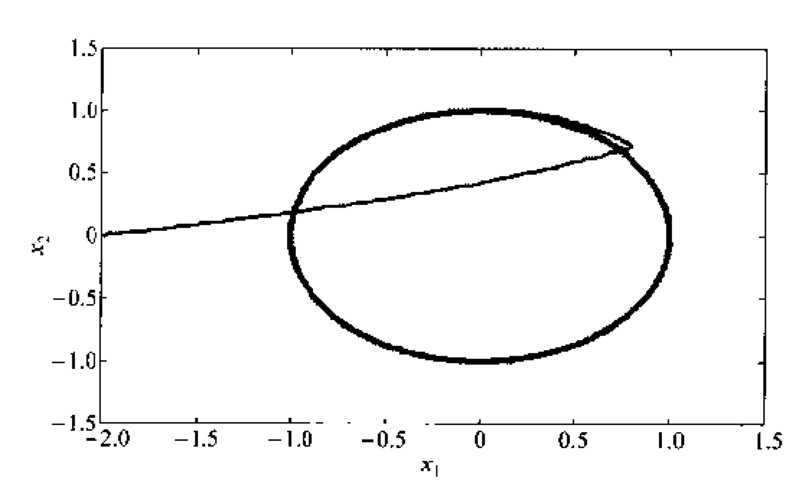


图 6-23　圆轨迹的位置跟踪

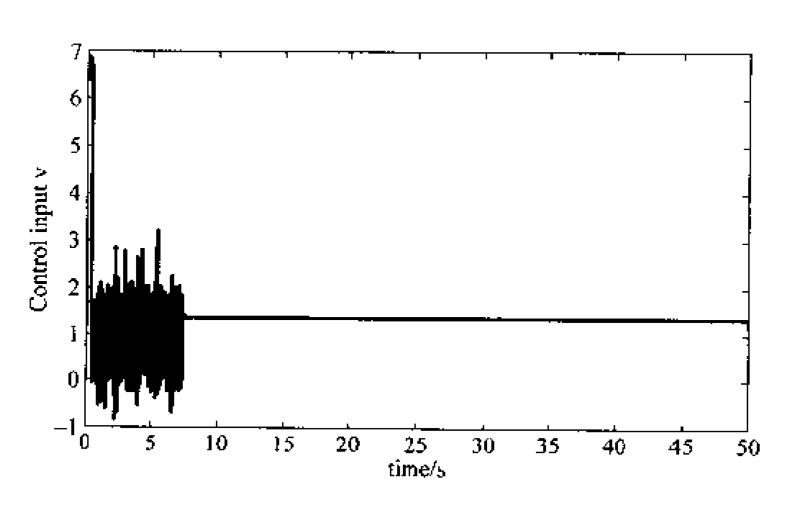


图 6-24　控制输入信号 v

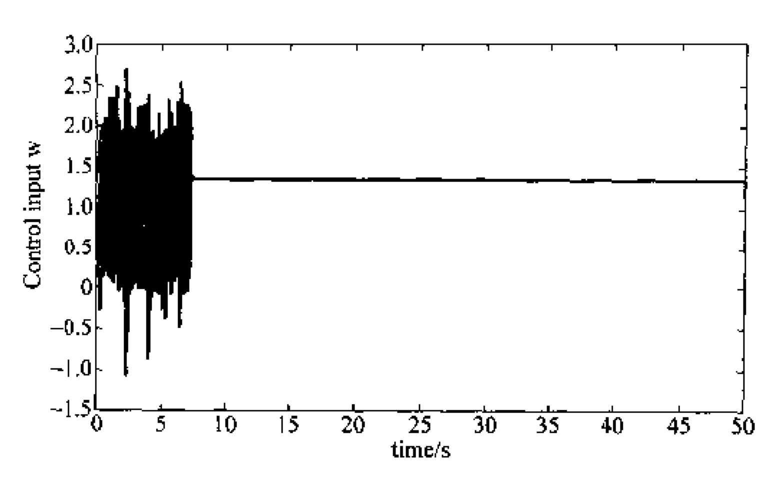


图 6-25 控制输入信号 ω

实例 2：

跟踪线速度为匀速运动、角速度为正弦运动的曲线轨迹。在主程序图中被控对象采用S函数 chap6_5plantn. m 来表示。取 $\omega_r=\sin t, v_r=1.0$，则 $\dot{v}_r=0$，位姿指令 $\boldsymbol{p}_r=(x_r \quad y_r \quad \theta_r)^T$ 的变化率为

$$\begin{aligned}\dot{x}_r &= v_r\cos\theta_r \\ \dot{y}_r &= v_r\sin\theta_r \\ \dot{\theta}_r &= \omega_r = \sin t\end{aligned}$$

通过解微分方程可求出位姿指令 $\boldsymbol{p}_r=(x_r \quad y_r \quad \theta_r)^T$。取 $\delta_1=\delta_2=0.02, k_1=k_2=6.0$，位姿误差初始值为[3 0 0]，采用控制律式(6.88)，仿真结果如图 6-26～图 6-32 所示。

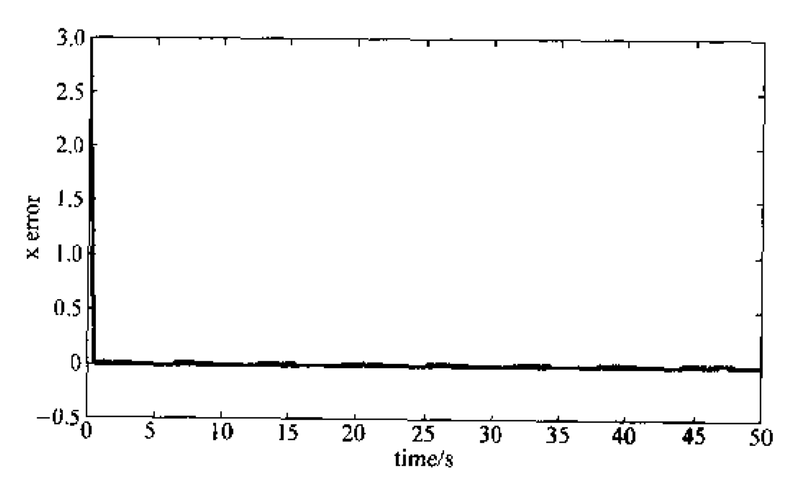


图 6-26 X 轴方向误差

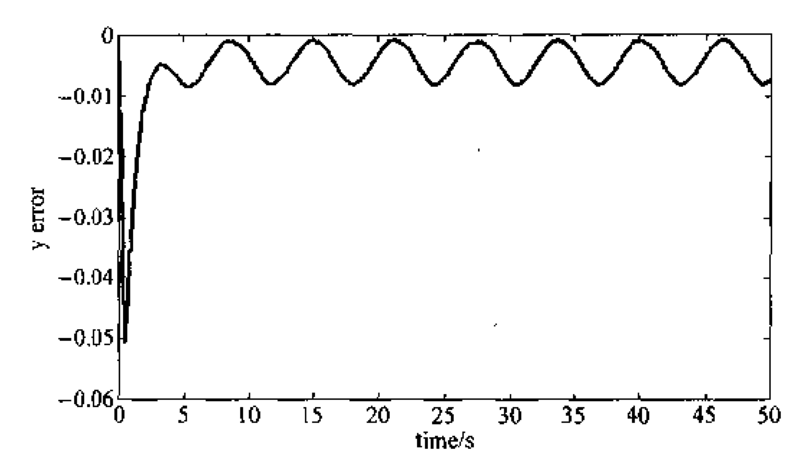


图 6-27　Y 轴方向误差

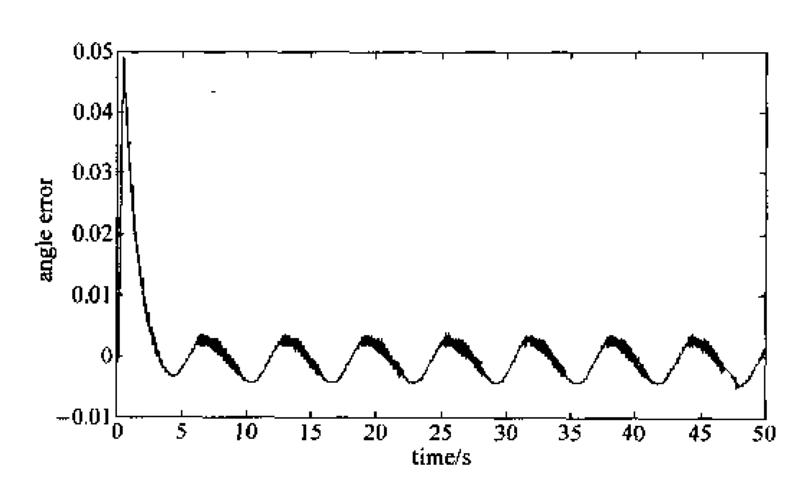


图 6-28　航向角方向误差

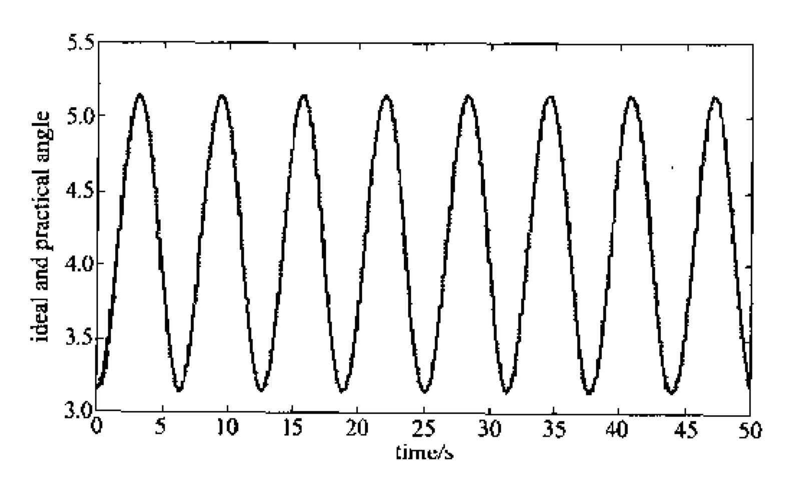


图 6-29　航向角的位置跟踪

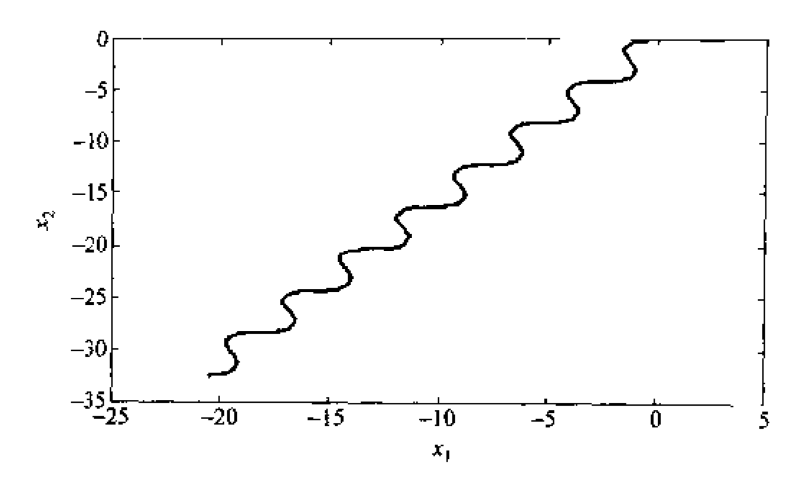


图 6-30 曲线轨迹的位置跟踪

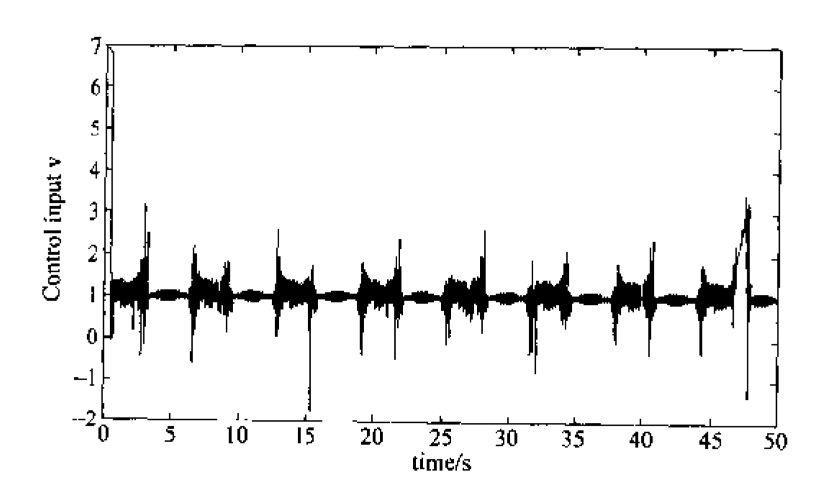


图 6-31 控制输入信号 v

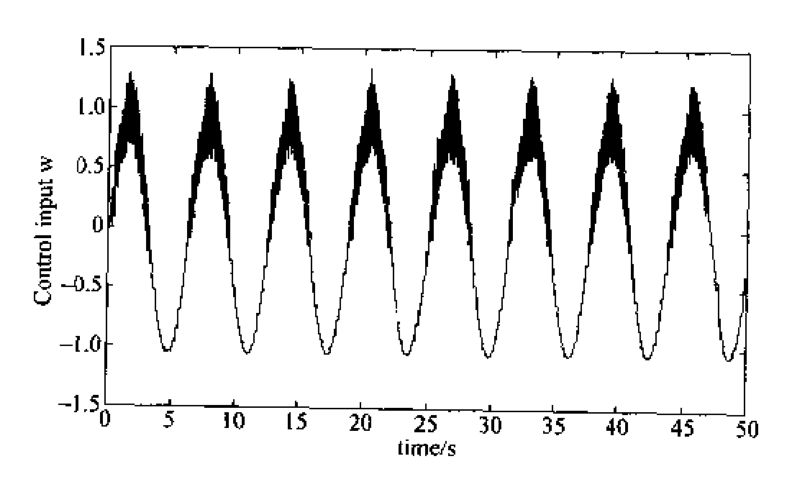


图 6-32 控制输入信号 w

仿真程序：

(1) Simulink 主程序：(如图 6-33 所示)chap6_5sim. mdl

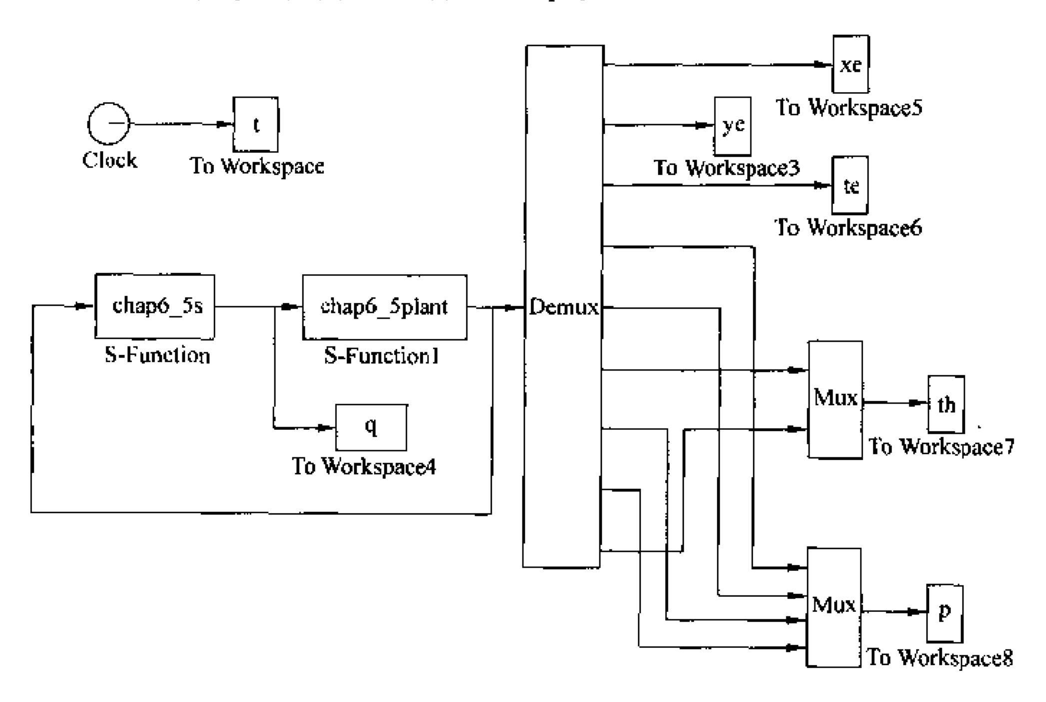


图 6-33　主程序图

(2) 控制器 S 函数程序：chap6_5s. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 9;
sizes.DirFeedthrough  = 1;
sizes.NumSampleTimes  = 1;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0 0];

function sys=mdlOutputs(t,x,u)
```

```
vr = 1.0;
wr = 1.0;

xe = u(1);
ye = u(2);
te = u(3);

delta1 = 0.02;
delta2 = 0.02;
k1 = 6.0;
k2 = 6.0;
s1 = xe;
s2 = te + atan(vr * ye);

Q = vr/(1 + vr^2 * ye^2);
q(2) = (wr + Q * vr * sin(te) + k2 * s2/(abs(s2) + delta2))/(1 + Q * xe);
w = q(2);
q(1) = ye * w + vr * cos(te) + k1 * s1/(abs(s1) + delta1);

sys(1) = q(1);
sys(2) = q(2);
```

实例 1 中被控对象 S 函数程序:chap6_5plant. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 3;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 9;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [3;0;0];
str = [];
ts  = [0 0];
```

```
function sys = mdlDerivatives(t,x,u)    % Time-varying model

xe = x(1);
ye = x(2);
the = x(3);

v = u(1);
w = u(2);

vr = 1.0;
wr = 1.0;

sys(1) = ye * w - v + vr * cos(the);
sys(2) = - xe * w + vr * sin(the);
sys(3) = wr - w;
function sys = mdlOutputs(t,x,u)
xe = x(1);
ye = x(2);
the = x(3);

vr = 1.0;
wr = 1.0;
r = vr/wr;

xr = r * cos(t);
yr = r * sin(t);
thr = wr * t;

th = thr - x(3);
M1 = [cos(th) sin(th); - sin(th) cos(th)];
M2 = [xr;yr] - inv(M1) * [xe;ye];
xp = M2(1);
yp = M2(2);

sys(1) = x(1);
sys(2) = x(2);
sys(3) = x(3);
sys(4) = xp;
sys(5) = yp;
sys(6) = th;
sys(7) = xr;
sys(8) = yr;
sys(9) = thr;
```

实例 2 中被控对象 S 函数程序:chap6_5plantn.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
```

```
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 6;
sizes.NumOutputs     = 9;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [3;0;0;0;0;pi];
str = [];
ts  = [0 0];

function sys = mdlDerivatives(t,x,u)

xe = x(1);
ye = x(2);
the = x(3);

v = u(1);
w = u(2);

vr = 1.0;
wr = sin(t);
thr = x(6);

sys(1) = ye * w - v + vr * cos(the);
sys(2) = - xe * w + vr * sin(the);
sys(3) = wr   w;

sys(4) = vr * cos(thr);
sys(5) = vr * sin(thr);
sys(6) = sin(t);

function sys = mdlOutputs(t,x,u)
xe = x(1);
ye = x(2);
the = x(3);

vr = 1.0;
wr = sin(t);
```

```
xr = x(4);
yr = x(5);
thr = x(6);

th = thr - x(3);
M1 = [cos(th) sin(th); - sin(th) cos(th)];
M2 = [xr;yr] - inv(M1) * [xe;ye];
xp = M2(1);
yp = M2(2);

sys(1) = x(1);
sys(2) = x(2);
sys(3) = x(3);
sys(4) = xp;
sys(5) = yp;
sys(6) = th;
sys(7) = xr;
sys(8) = yr;
sys(9) = thr;
```

作图程序:chap6_5plot.m

```
close all;

figure(1);
plot(t,xe,'r');
xlabel('time(s)');ylabel('x error');

figure(2);
plot(t,ye,'r');
xlabel('time(s)');ylabel('y error');

figure(3);
plot(t,te,'r');
xlabel('time(s)');ylabel('angle error');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(4);
plot(t,th(:,1),'b');
hold on;
plot(t,th(:,2),'r');
xlabel('time(s)');ylabel('ideal and practical angle');

figure(5);
plot(P(:,3),P(:,4),'r');
xlabel('xr');ylabel('yr');

hold on;
plot(P(:,1),P(:,2),'b');
xlabel('x1');ylabel('x2');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(6);
plot(t,q(:,1),'r');
xlabel('time(s)');ylabel('Control input v');

figure(7);
plot(t,q(:,2),'r');
xlabel('time(s)');ylabel('Control input w');
```

6.5 一种简单的移动机器人变结构控制

6.5.1 控制器设计

6.4 节中控制器的设计比较复杂。下面介绍一种简单的移动机器人控制律设计方法。移动机器人模型同 6.4。

取切换函数

$$s = \theta_e \tag{6.89}$$

取等速趋近律，令

$$\dot{s} = -k_2 \operatorname{sgn} s \tag{6.90}$$

其中 $k_2>0$。

设计控制律为

$$q = \begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{bmatrix} v_r\cos\theta_e + k_1 x_e + \dfrac{y_e}{x_e} v_r \sin\theta_e \\ \omega_r + k_2 \operatorname{sgn} s \end{bmatrix} \tag{6.91}$$

其中 $k_1>0$。

6.5.2 稳定性分析

设计 Lyapunov 函数为

$$V = \frac{1}{2}x_e^2 + \frac{1}{2}y_e^2 + \frac{1}{2}\theta_e^2 \tag{6.92}$$

则

$$\begin{aligned}
\dot{V} &= x_e\dot{x}_e + y_e\dot{y}_e + \theta_e\dot{\theta}_e = x_e(y_e\omega - v + v_r\cos\theta_e) + y_e(-x_e\omega + v_r\sin\theta_e) + s\dot{s} \\
&= x_e\left(y_e\omega - v_r\cos\theta_e - k_1 x_e - \frac{y_e}{x_e}v_r\sin\theta_e + v_r\cos\theta_e\right) + y_e(-x_e\omega + v_r\sin\theta_e) - k_2 \mid s \mid \\
&= -k_1 x_e^2 - k_2 \mid s \mid \leqslant 0
\end{aligned}$$

6.5.3 仿真实例

针对 6.4.4 节中仿真实例 1 进行仿真。仿真的 Simulink 主程序如图 6-33 所示，被控对象

程序、作图程序与 6.4.4 节中的仿真实例 1 相同，将控制律的 S 函数程序改为 chap6_5sn.m。采用控制律式(6.91)，取 $k_1=k_2=10$，圆轨迹的位置跟踪结果如图 6-34 所示。

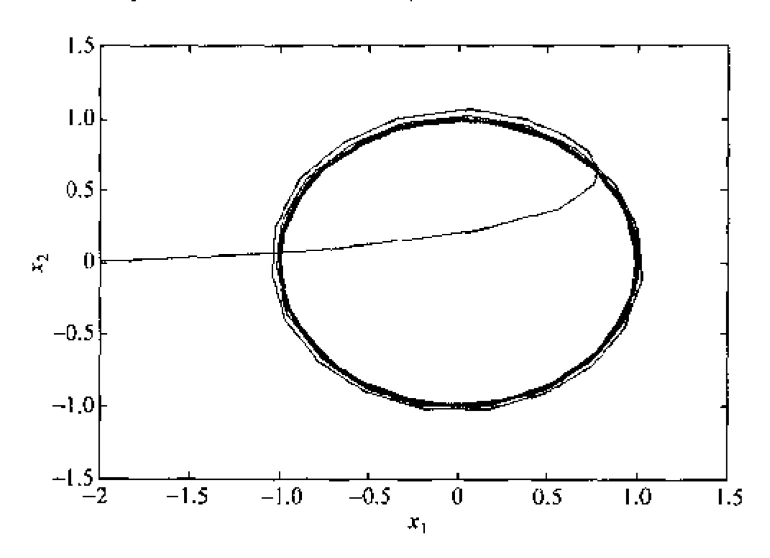


图 6-34　圆轨迹的位置跟踪

控制器 S 函数程序：chap6_5sn.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 9;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0 0];

function sys = mdlOutputs(t,x,u)
vr = 1.0;
wr = 1.0;

xe = u(1);
```

```
ye = u(2);
te = u(3);

k1 = 10;
k2 = 10;

q(2) = wr + k2 * sign(te);
w = q(2);
q(1) = vr * cos(te) + k1 * xe + ye/(xe + 0.01) * vr * sin(te);

sys(1) = q(1);
sys(2) = q(2);
```

参考文献

1. Lin F J, Shen P H, Hsu S P. Adaptive backstepping sliding mode control for linear induction motor drive. IEE Proceeding Electrical Power Application, 2002, 149(3): 184~194
2. Tan Y L, Chang J, Tan H L, Hu J. Integral backstepping control and experimental implementation for motion system. Proceedings of the 2000 IEEE International Conference on Control Applications, Anchorage, Alaska, USA, September pp. 25~27
3. Kanayama Y, Kimura Y, Miyazaki F, et al. A stable tracking control method for autonomous mobile robot. IEEE International Conference on Robotics and Automation, 1990, 384~389
4. 叶涛,侯增广,谭民,李磊,陈细军. 移动机器人的滑模轨迹跟踪控制.高技术通讯,2004,1:71~74
5. 吴卫国,陈辉堂,王月娟. 移动机器人的全局轨迹跟踪控制.自动化学报,2001,27(3):326~331
6. Shim H S, Kim J H, Koh K. Variable structure control of nonholonomic wheeled mobile robot. IEEE International Conference on Robotics and Automation, 1995, 1694~1699

第 7 章　动态滑模控制

动态滑模方法通过设计新的切换函数或将常规滑模变结构控制中的切换函数 s 通过微分环节构成新的切换函数 σ，该切换函数与系统控制输入的一阶或高阶导数有关，可将不连续项转移到控制的一阶或高阶导数中去，得到在时间上本质连续的动态滑模控制律，有效地降低了抖振。

7.1　一阶动态滑模控制

设计新的切换函数 s，使该切换函数与系统控制输入的一阶导数有关，从而将不连续项转移到控制的一阶导数中去，得到在时间上本质连续的动态滑模控制律，有效地降低抖振[1]。

7.1.1　控制器的设计

考虑如下二阶系统：

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}(u + d) \tag{7.1}$$

其中 $\boldsymbol{x}=[x_1 \quad x_2]^{\mathrm{T}}$，$u\in\mathbf{R}$ 为控制输入，$d\in\mathbf{R}$ 为外加干扰。

设位置指令为

$$r = A\sin(\omega t), \dot{r} = \omega A\cos(\omega t) \tag{7.2}$$

则误差及导数为

$$\boldsymbol{e} = [r - x_1 \quad \dot{r} - x_2]^{\mathrm{T}} = \boldsymbol{R} - \boldsymbol{x} \tag{7.3}$$

其中 $R=[r \quad \dot{r}]^{\mathrm{T}}$。

定义切换函数为

$$s = \boldsymbol{C}\boldsymbol{e} + Du \tag{7.4}$$

其中 $D>0$，$\boldsymbol{C}=[C(1) \quad C(2)]$。当 $s=0$ 时，$Du=-\boldsymbol{C}\boldsymbol{e}$。为了形成负反馈，要求 $\boldsymbol{C}<0$。则切换函数的一阶导数为

$$\dot{s} = \boldsymbol{C}\dot{\boldsymbol{R}} - \boldsymbol{C}\dot{\boldsymbol{x}} + D\dot{u} \tag{7.5}$$

采用等速趋近律：

$$\dot{s} = -\eta\operatorname{sgn}(s) \tag{7.6}$$

则由式(7.5)、式(7.6)得到动态控制律为

$$\dot{u} = \frac{1}{D}(-\boldsymbol{C}\dot{\boldsymbol{R}} + \boldsymbol{C}\dot{\boldsymbol{x}} - \eta\operatorname{sgn}(s)) \tag{7.7}$$

7.1.2　仿真实例

考虑如下二阶线性时变系统：

$$\dot{x} = Ax + B(u + d)$$

其中 $A=\begin{bmatrix}0 & 1\\ 2 & 3\end{bmatrix}$，$B=\begin{bmatrix}0\\ 1\end{bmatrix}$。

设期望的跟踪信号为 $r=0.5\sin(\pi t)$。外加干扰为 $d=\sin(2\pi t)$，初始条件为 $x(0)=[0\quad 0]$。动态滑模控制律中，取 $C=[-2000\quad -10]$，$D=0.5$，$\gamma=15$。仿真结果如图 7-1～图 7-3 所示。

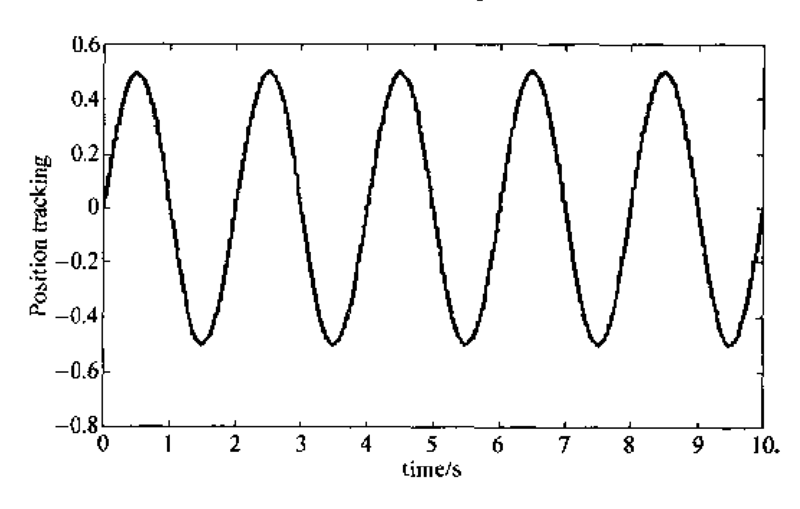


图 7-1　位置跟踪

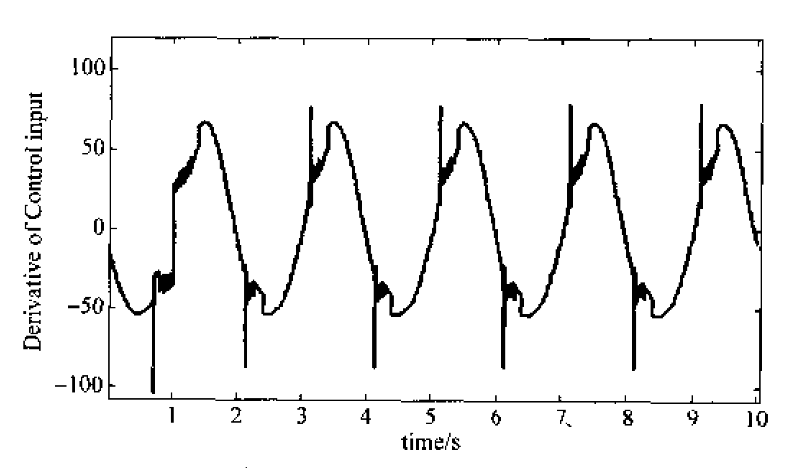


图 7-2　动态控制输入信号 $\dot{u}$

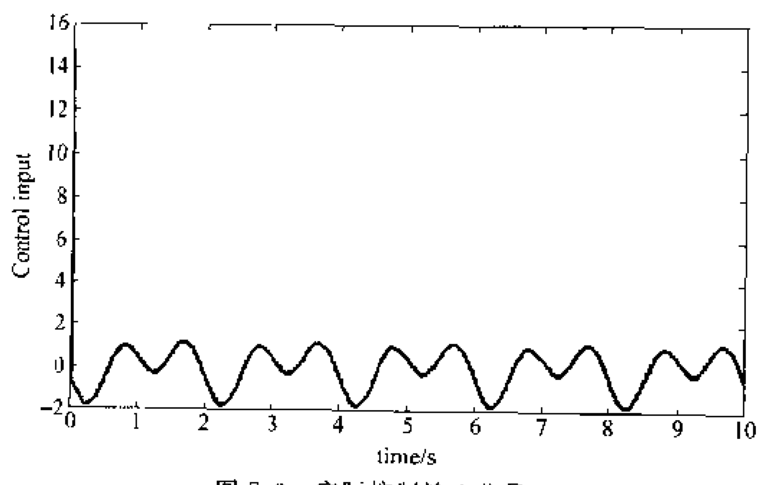


图 7-3　实际控制输入信号 u

仿真程序：

(1) 主程序(如图 7-4 所示)：chap7_1sim. mdl

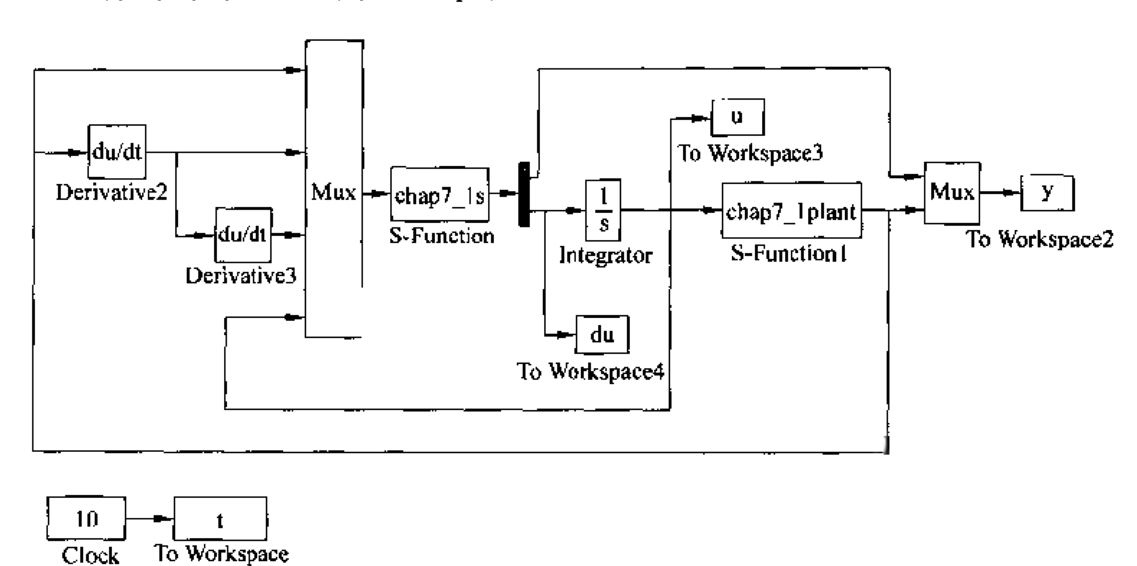


图 7-4 主程序图

(2) S 函数控制子程序：chap7_1s. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [];

function sys = mdlOutputs(t,x,u)
x1 = u(1);
dx1 = u(2);
```

```
ddx1 = u(3);
ut = u(4);

r = 0.5 * sin(pi * t);
dr = 0.5 * pi * cos(pi * t);
ddr = - 0.5 * pi^2 * sin(pi * t);

R = [r;dr];
dR = [dr;ddr];

C = - [2000 10];
D = 0.5;

xx = [x1;dx1];
s = C * R - C * xx + D * ut;

dxx = [dx1;ddx1];
xite = 15;
du = 1/D * ( - C * dR + C * dxx - xite * sign(s));

sys(1) = r;
sys(2) = du;
```

(3) 被控对象子程序:chap7_1plant.m

```
% S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 2;
```

```
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 1;
sizes.NumInputs       = 1;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 0;

sys = simsizes(sizes);
x0 = [0,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
A = [0 1;2 3];
B = [0;10];

sys(1) = x(2);
sys(2) = A(2,1) * x(1) + A(2,2) * x(2) + B(2) * (u + sin(2 * pi * t));
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
```

(4) 作图程序：chap7_1plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,du(:,1),'r');
xlabel('time(s)');ylabel('Derivative of Control input');

figure(3);
plot(t,u,'r');
xlabel('time(s)');ylabel('Control input');
```

7.2　基于动态切换函数的动态滑模控制

7.2.1　控制器的设计

考虑如下单入单出 n 阶仿射非线性系统：

$$\begin{cases}\dot{x}_i = x_{i+1}, & i = 1,2,\cdots,n-1 \\ \dot{x}_n = f(x) + g(x)u + \eta \\ y = x_1\end{cases} \tag{7.8}$$

其中 $x \in \mathbf{R}$ 为可测状态变量，$u, y \in \mathbf{R}$ 分别为系统的输入和输出，$f(x)$ 和 $g(x)$ 为已知平滑函

数，η 为系统中的不确定项，它包括模型不确定性和外加扰动。

定义误差及切换函数分别为

$$\begin{cases} e = y - y_d \\ s = c_1 e_1 + c_2 e_2 + \cdots + c_{n-1} e_{n-1} + e_n = \sum_{i=1}^{n-1} c_i e_i + e_n \end{cases} \tag{7.9}$$

其中 $e_i = e^{(i-1)} (i=1,2,\cdots,n)$ 为跟踪误差及其各阶导数，选取常数 $c_1, c_2, \cdots, c_{n-1}$，使得多项式 $p^{n-1} + c_{n-1} p^{n-2} + \cdots + c_2 p + c_1$ 为 Hurwitz（霍尔伍兹）稳定，p 为 Laplace（拉普拉斯）算子。则

$$\dot{s} = f(\boldsymbol{x}) + g(\boldsymbol{x}) u + \eta - y_d^{(n)} + \sum_{i=1}^{n-1} c_i e_{i+1} \tag{7.10}$$

构造新的动态切换函数

$$\sigma = \dot{s} + \lambda s \tag{7.11}$$

其中 λ 为严格的正常数。

当 $\sigma = 0$ 时，$\dot{s} + \lambda s = 0$ 是一个渐近稳定的一阶动态系统，s 趋近于零。

假设 1 不确定性满足有界条件，存在有界函数 $B_n(\boldsymbol{x})$，使得

$$|\eta| \leqslant B_n(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathbf{R}^n$$

且 $g(x)$ 符号恒定。

假设 2 不确定项导数有界

$$|\dot{\eta}| \leqslant \bar{B}_n(x) \quad \forall \boldsymbol{x} \in \mathbf{R}^n \tag{7.12}$$

假设 3 存在正实数 ε，满足

$$\varepsilon > (c_{n-1} + \lambda) B_n + \bar{B}_n \tag{7.13}$$

动态滑模控制律取为

$$\dot{u} = \frac{1}{g} \Big[-\Big((c_{n-1} + \lambda) g + \frac{\mathrm{d}g}{\mathrm{d}x} \dot{x} \Big) u - (c_{n-1} + \lambda) f - \frac{\mathrm{d}f}{\mathrm{d}x} \dot{x} + (c_{n-1} + \lambda) y_d^{(n)} + y_d^{(n+1)} - \sum_{i=1}^{n-2} c_i e_{i+2} - \sum_{i=1}^{n-1} \lambda c_i e_{i+1} - \varepsilon \operatorname{sgn}(\sigma) \Big] \tag{7.14}$$

稳定性分析：

将式(7.9)代入式(7.11)得

$$\sigma = \sum_{i=1}^{n-1} c_i e_{i+1} + \dot{e}_n + \lambda \Big(\sum_{i=1}^{n-1} c_i e_i + e_n \Big) \tag{7.15}$$

则

$$\dot{\sigma} = \sum_{i=1}^{n-2} c_i e_{i+2} + c_{n-1} \dot{e}_n + \ddot{e}_n + \lambda \Big(\sum_{i=1}^{n-1} c_i e_{i+1} + \dot{e}_n \Big) \tag{7.16}$$

将式(7.8)和式(7.9)代入式(7.16)整理得

$$\dot{\sigma} = \sum_{i=1}^{n-2} c_i e_{i+2} + \sum_{i=1}^{n-1} \lambda c_i e_{i+1} + (c_{n-1} + \lambda) f + \frac{\mathrm{d}f}{\mathrm{d}x} \dot{x} - (c_{n-1} + \lambda) y_d^{(n)} - y_d^{(n+1)} + \Big[(c_{n-1} + \lambda) g + \frac{\mathrm{d}g}{\mathrm{d}x} \dot{x} \Big] u + g \dot{u} + (c_{n-1} + \lambda) \eta + \dot{\eta} \tag{7.17}$$

将控制律式(7.14)代入式(7.17)中，得

$$\dot{\sigma} = (c_{n-1} + \lambda) \eta + \dot{\eta} - \varepsilon \operatorname{sgn}(\sigma)$$

则根据假设1～3得

$$
\begin{aligned}
\sigma\dot{\sigma} &= \sigma(c_{n-1}+\lambda)\eta+\sigma\dot{\eta}-\varepsilon\mid\sigma\mid=\sigma[(c_{n-1}+\lambda)\eta+\dot{\eta}]-\varepsilon\mid\sigma\mid \\
&< \sigma[(c_{n-1}+\lambda)\eta+\dot{\eta}]-[(c_{n-1}+\lambda)B_n+\bar{B}_n]\mid\sigma\mid\leqslant 0
\end{aligned}
$$

7.2.2 仿真实例

考虑如下二阶线性时变系统：

$$
\begin{cases}
\dot{x}_1 = x_2 \\
\dot{x}_2 = -25x_2+133u+1.5\sin(t)
\end{cases}
$$

其中 $f(\boldsymbol{x})=-25x_2$，$g(\boldsymbol{x})=133$。

设期望的跟踪信号为 $r=\sin(0.2t)+0.5\cos(t)$，跟踪误差为 $e=\boldsymbol{x}(1)-r$。$n=2$ 时，定义 $s=5e+\dot{e}$，即 $c_1=5$，取 $\lambda=15$，初始条件为 $\boldsymbol{x}(0)=[0.5\quad 0]$。动态滑模控制律为 $\dot{u}=\frac{1}{g}[-(c_1+\lambda)gu-(c_1+\lambda)f-\dot{f}+(c_1-\lambda)\ddot{r}+\dddot{r}-\lambda c_1\dot{e}-\varepsilon\mathrm{sgn}(\sigma)]$，$\varepsilon$ 按式(7.13)取值，取 $\varepsilon=(c_1+\lambda)B_n+\bar{B}_n+2.0$，$B_n=1.5$，$\bar{B}_n=1.5$，仿真结果如图7-5～图7-7所示。

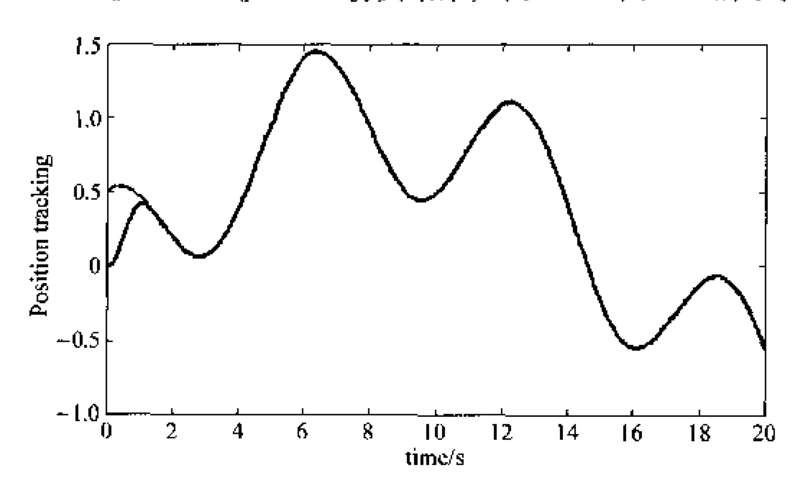


图7-5　位置跟踪

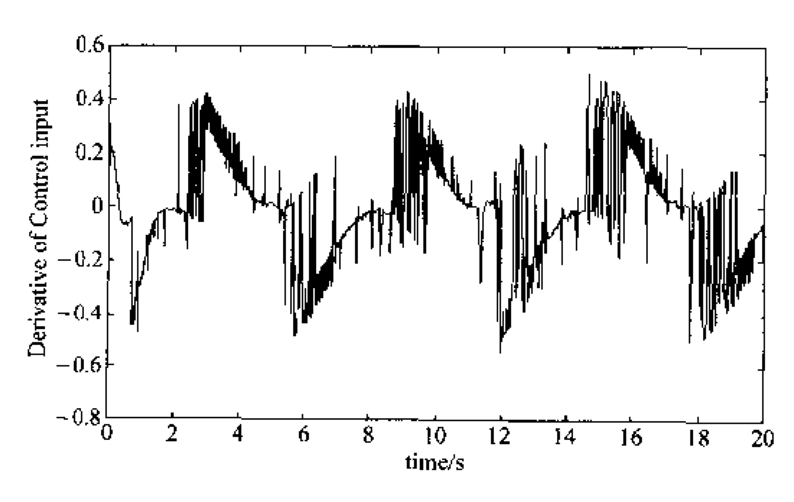


图7-6　动态控制输入信号 $\dot{u}$

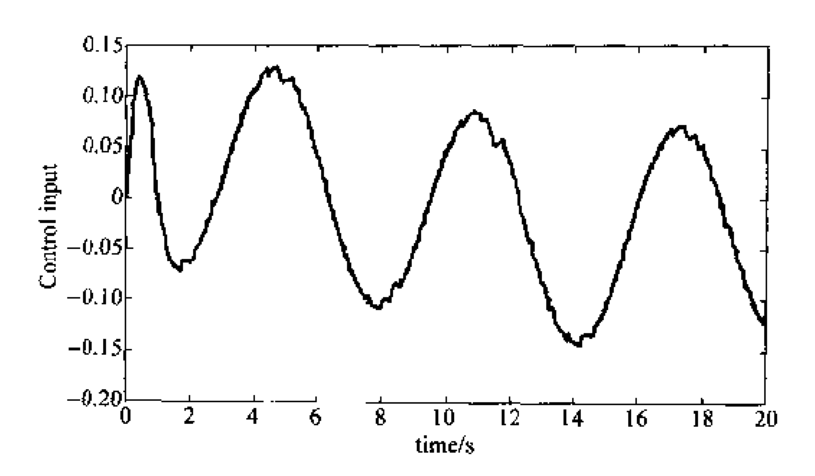


图 7-7　实际控制输入信号 u

仿真程序：

(1) 主程序(如图 7-8 所示)：chap7_2sim.mdl

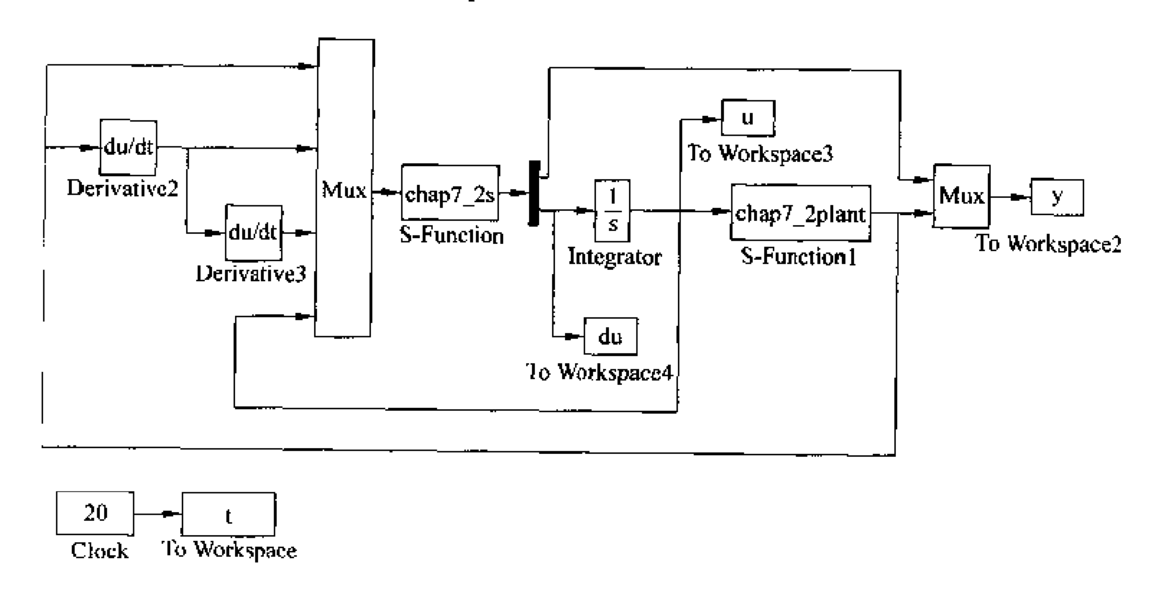


图 7-8　主程序图

(2) S 函数控制子程序：chap7_2s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
```

```
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [];

function sys = mdlOutputs(t,x,u)
x1 = u(1);
dx1 = u(2);
ddx1 = u(3);
ut = u(4);

r = sin(0.2 * t) + 0.5 * cos(t);
dr = 0.2 * cos(0.2 * t) - 0.5 * sin(t);
ddr = - 0.2^2 * sin(0.2 * t) - 0.5 * cos(t);
dddr = - 0.2^3 * cos(0.2 * t) + 0.5 * sin(t);

e = x1 - r;
de = dx1 - dr;
dde = ddx1 - ddr;

c = 5;
s = c * e + de;

fx = - 25 * dx1;
df = - 25 * ddx1;

gx = 133;
nmn = 15;

Dmax = 1.5;
dDmax = 1.5;
ec = (c + nmn) * Dmax + dDmax + 2.0;

ds = c * de + dde;
rou = ds + nmn * s;
du = 1/gx * [ - (c + nmn) * 133 * ut - (c + nmn) * fx - df + (c + nmn) * ddr + dddr - nmn * c * de - ec *
sign(rou)];

sys(1) = r;
sys(2) = du;
```

(3) 被控对象子程序：chap7_2plant.m

```
%S-function for continuous state equation
function [sys,x0,str,ts]=s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys=mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys=simsizes(sizes);
x0=[0,0];
str=[];
ts=[];

function sys=mdlDerivatives(t,x,u)
Dmax=1.5;
D=Dmax*sin(t);
dDmax=1.5;

sys(1)=x(2);
sys(2)=-25*x(2)+133*u+D;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
```

(4) 作图程序:chap7_2plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
```

```
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,du(:,1),'r');
xlabel('time(s)');ylabel('Derivative of Control input');

figure(3);
plot(t,u,'r');
xlabel('time(s)');ylabel('Control input');
```

7.3　基于反演设计的自适应动态滑模控制

利用自适应技术与backstepping控制方法相结合，通过设计新型切换函数，采用一阶动态滑模控制方法，可实现不确定系统的无抖振滑模控制[2]。

考虑如下一阶不确定性系统：

$$\dot{x} = u + \theta x^2 \tag{7.18}$$

其中θ为未知常数。

7.3.1　常规自适应滑模控制器

针对系统式(7.18)，将切换函数设计为

$$s = x \tag{7.19}$$

设$\hat{\theta}$为θ的估计值，控制律设计为

$$u = -k\text{sgn}(s) - \hat{\theta}s^2 \tag{7.20}$$

稳定性分析：

定义Lyapunov函数

$$V = \frac{1}{2}s^2 + \frac{1}{2}\phi^2$$

其中ϕ为θ的估计误差，$\phi=\theta-\hat{\theta}$。则

$$\dot{V} = s\dot{s} + \phi\dot{\phi} = x\dot{x} - (\theta - \hat{\theta})\dot{\hat{\theta}} = x(u + \theta x^2) - \theta\dot{\hat{\theta}} + \hat{\theta}\dot{\hat{\theta}}$$

设计自适应律为

$$\dot{\hat{\theta}} = x^3 \tag{7.21}$$

则

$$\dot{V} = xu + \hat{\theta}x^3 = x(-k\text{sgn}(x) - \hat{\theta}x^2) + \hat{\theta}x^3 = -k\,|\,x\,|$$

7.3.2　仿真实例

针对不确定性系统式(7.18)，取$\theta=1$，系统初始状态为$x=0.5$。采用滑模控制律式(7.20)，仿真结果如图7-9和图7-10所示。可见，由于控制器中包含有切换项，控制输入信

号带有很大抖振。

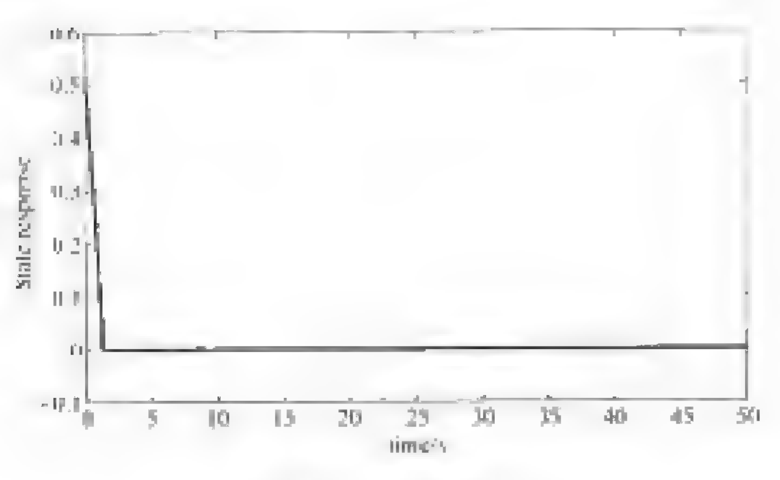


图 7-9　状态响应

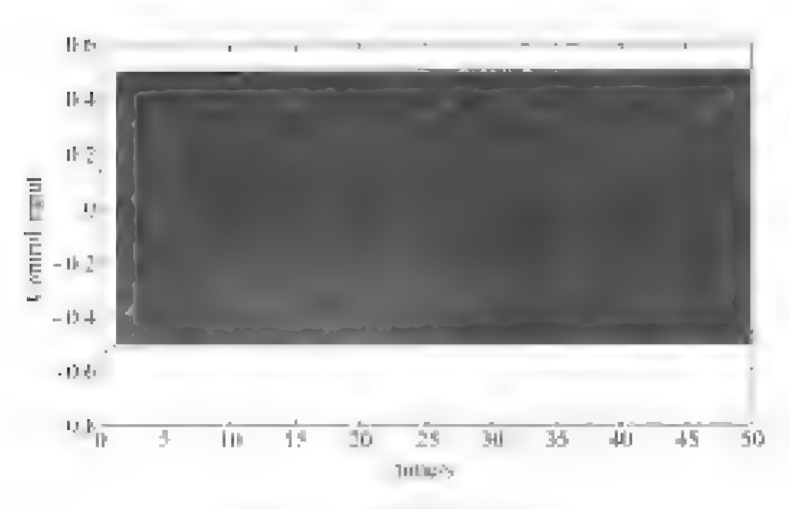


图 7-10　控制输入信号 u

仿真程序：

(1) 主程序（如图 7-11 所示）：chap7_3sim.mdl

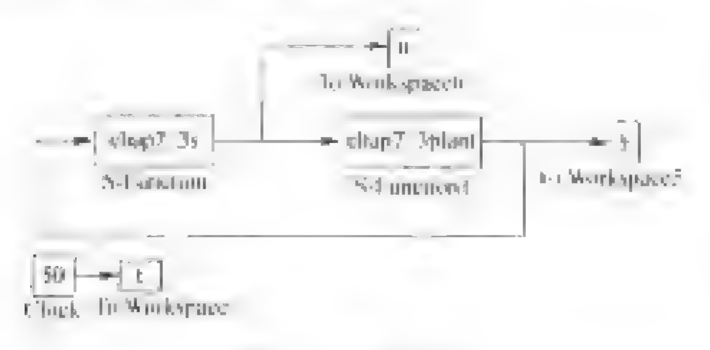


图 7-11　主程序图

(2) S 函数子程序：chap7_3s.m

```
%S-function for continuous state equation
```

```
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
  case 1,
    sys=mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys=mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys=simsizes(sizes);
x0=[0.15];
str=[];
ts=[];
function sys=mdlDerivatives(t,x,u)
sys(1)=u(1)^3;

function sys=mdlOutputs(t,x,u)
k=0.5;
sys(1)=-k*sign(u(1))-x(1)*u(1)^2;
```

(3) 被控对象子程序:chap7_3plant.m

```
%S-function for continuous state equation
function [sys,x0,str,ts]=s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
```

```
case 1,
    sys = mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0.5];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
th = 1;

sys(1) = u + th * x(1)^2;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
```

(4) 作图子程序:chap7_3plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r');
xlabel('time(s)');ylabel('state response');

figure(2);
plot(t,u,'r');
xlabel('time(s)');ylabel('control input');
```

7.3.3 反演自适应动态滑模控制

由于常规滑模控制器中带有切换项,容易造成抖振。采用动态滑模控制的方法,可有效

地降低抖振。

一阶系统式(7.18)可写为如下形式：

$$\dot{x} = u + \theta x^2 \tag{7.22}$$

$$\dot{u} = v \tag{7.23}$$

其中 v 为辅助项，用于设计动态控制律。

切换函数设计为

$$s = c_1 x + u + \hat{\theta}_1 x^2 \tag{7.24}$$

其中 $c_1 > 0$，$\hat{\theta}_1$ 为未知项 θ 的第一个估计值。

由式(7.22)和式(7.24)得

$$\dot{x} = s - c_1 x - \hat{\theta}_1 x^2 + \theta x^2 = -c_1 x + s + (\theta - \hat{\theta}_1)x^2 \tag{7.25}$$

定义 Lyapunov 函数

$$V_1 = \frac{1}{2}x^2 + \frac{1}{2}(\theta - \hat{\theta}_1)^2 \tag{7.26}$$

则

$$\begin{aligned}\dot{V}_1 &= x\dot{x} + (\theta - \hat{\theta}_1)(-\dot{\hat{\theta}}_1) = x(s - c_1 x - \hat{\theta}_1 x^2 + \theta x^2) - (\theta - \hat{\theta}_1)\dot{\hat{\theta}}_1 \\ &= -c_1 x^2 + xs + (\theta - \hat{\theta}_1)(x^3 - \dot{\hat{\theta}}_1)\end{aligned}$$

设计自适应律为

$$\dot{\hat{\theta}}_1 = x^3 \tag{7.27}$$

则

$$\dot{V}_1 = -c_1 x^2 + xs \tag{7.28}$$

由式(7.24)可知

$$\begin{aligned}\dot{s} &= c_1\dot{x} + \dot{u} + \dot{\hat{\theta}}_1 x^2 + 2\dot{x}\hat{\theta}_1 x = v + (c_1 + 2\hat{\theta}_1 x)\dot{x} + x^5 \\ &= v + (c_1 + 2\hat{\theta}_1 x)[-c_1 x + s + (\theta - \hat{\theta}_1)x^2] + x^5 \\ &= v + (c_1 + 2\hat{\theta}_1 x)(-c_1 x + s - \hat{\theta}_1 x^2) + x^5 + (c_1 + 2\hat{\theta}_1 x)\theta x^2 \\ &= v + \psi_1 + \theta\psi_2\end{aligned}$$

其中 $\psi_1 = (c_1 + 2\hat{\theta}_1 x)(-c_1 x + s - \hat{\theta}_1 x^2) + x^5 - (c_1 + 2\hat{\theta}_1 x)u + x^5$，$\psi_2 = (c_1 + 2\hat{\theta}_1 x)x^2$。

令动态滑模控制律为

$$v = \dot{u} = -\psi_1 - \hat{\theta}_2\psi_2 - x - k\,\mathrm{sgn}(s) \tag{7.29}$$

则

$$\dot{s} = -x + (\theta - \hat{\theta}_2)\psi_2 - k\,\mathrm{sgn}(s)$$

其中 $\hat{\theta}_2$ 为未知参数 θ 的第二个估计值。

Lyapunov 函数设计为

$$V_2 = V_1 + \frac{1}{2}s^2 + \frac{1}{2}(\theta - \hat{\theta}_2)^2 \tag{7.30}$$

则

$$\begin{aligned}\dot{V}_2 &= \dot{V}_1 + s\dot{s} + (\theta - \hat{\theta}_2)(-\dot{\hat{\theta}}_2) = -c_1 x^2 + xs + s[-x + (\theta - \hat{\theta}_2)\psi_2 - k\,\mathrm{sgn}(s)] - \dot{\hat{\theta}}_2(\theta - \hat{\theta}_2) \\ &= -c_1 x^2 + s(\theta - \hat{\theta}_2)\psi_2 - k|s| - \dot{\hat{\theta}}_2(\theta - \hat{\theta}_2)\end{aligned}$$

令自适应律为

$$\dot{\hat{\theta}} = \gamma\sigma = \gamma(\dot{x}_1 + 2\bar{x}_1) \tag{7.31}$$

则

$$\dot{V}_1 = -c_1 x^2 - k|s| \leq 0$$

动态自适应滑模控制律由式(7.27)、式(7.29)和式(7.31)组成。

7.3.4　仿真实例

针对不确定性系统式(7.58)，取 $b=1$，系统初始状态为 $x(0)=0.5$，采用动态滑模控制律式(7.29)，取 $c=2.5$，和 $\hat{\theta}$ 的初始值取$(0\ 0)$，仿真结果如图7.12～图7.13所示。

可见，采用自适应动态滑模控制，可从根本上消除滑模控制的抖振。

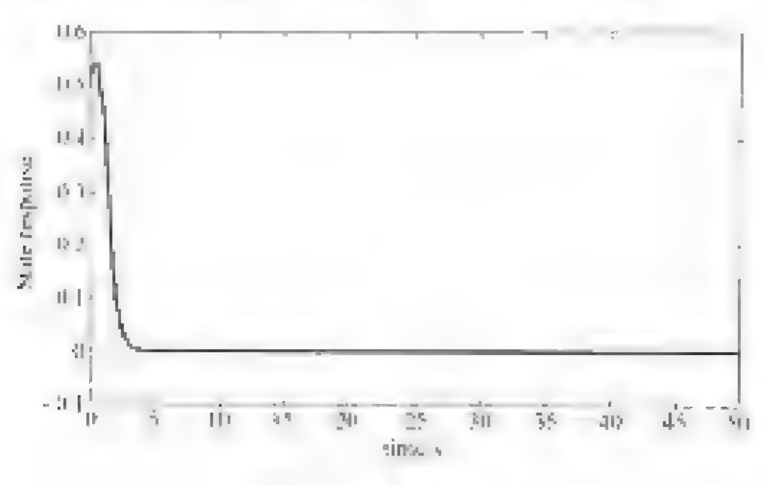


图7.12　状态响应

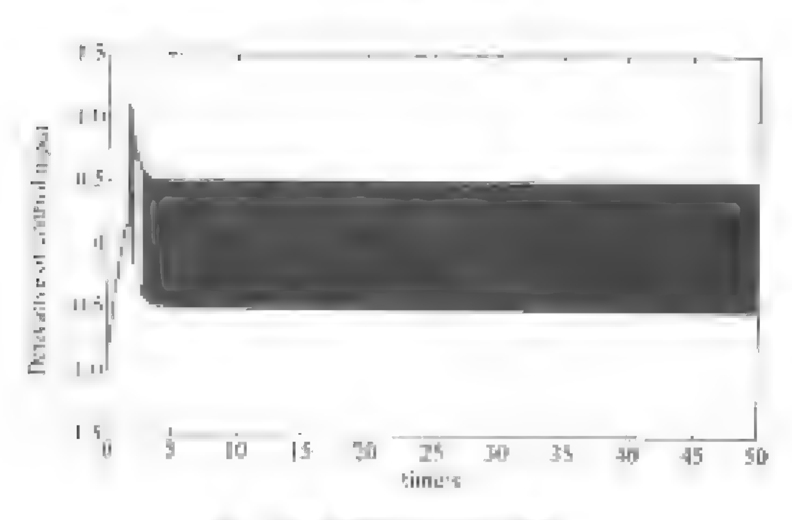


图7.13　控制输入信号的导数 $\dot{u}$

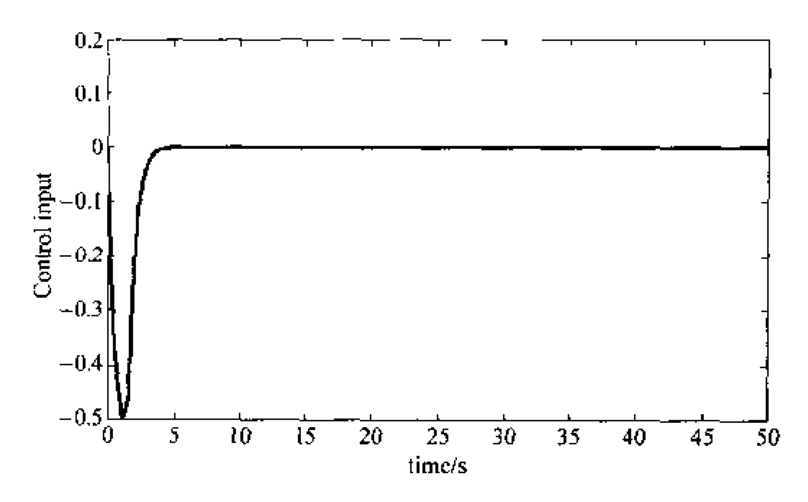


图 7-14　控制输入信号 u

仿真程序：

(1) 主程序(如图 7-15 所示)：chap7_4sim.mdl

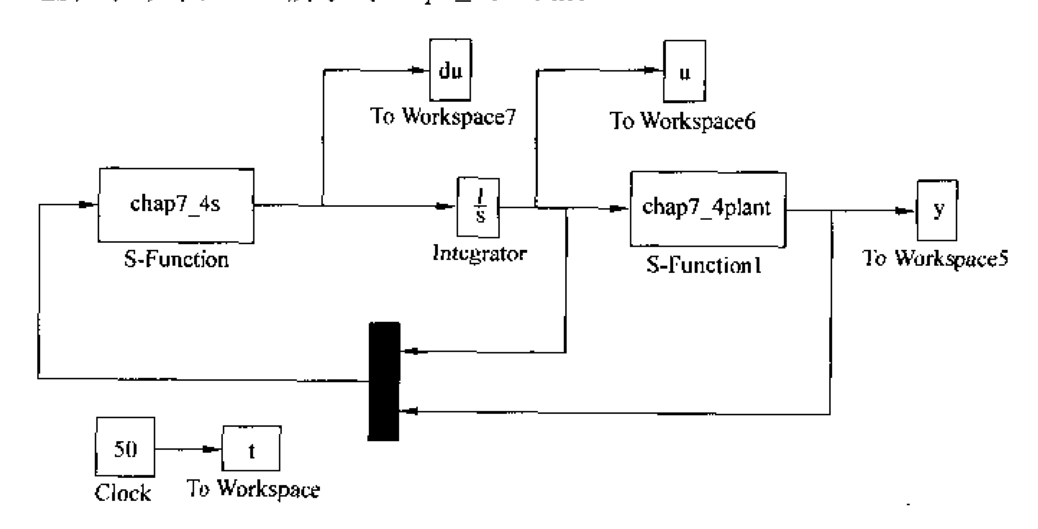


图 7-15　主程序图

(2) S 函数子程序：chap7_4s.m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
  case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
```

```
    case {2, 4, 9 }
        sys = [];
  % Unexpected flags
    otherwise
        error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 2;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 1;
sizes.NumInputs       = 2;
sizes.DirFeedthrough  = 1;
sizes.NumSampleTimes  = 0;

sys = simsizes(sizes);
x0 = [0,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
c1 = 2;
th1 = x(1);
s = c1 * u(2) + u(1) + x(1) * u(2)^2;

sys(1) = u(2)^3;
sys(2) = s * u(2)^2 * (c1 + 2 * th1 * u(2));
function sys = mdlOutputs(t,x,u)
k = 0.5;
c1 = 2;

th1 = x(1);
th2 = x(2);
s = c1 * u(2) + u(1) + th1 * u(2)^2;

sys(1) = - (c1 + 2 * th1 * u(2)) * u(1) - u(2)^5 - th2 * (c1 + 2 * th1 * u(2)) * u(2)^2 - u(2) - k *
sign(s);
```

(3) 被控对象子程序:chap7_4plant.m

```
% S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
```

```
case 1,
    sys = mdlDerivatives(t,x,u);
 % Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
 % Unhandled flags
  case {2, 4, 9 }
    sys = [];
 % Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

 % mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0.5];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
th = 1;
sys(1) = u + th * x(1)^2;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
```

(4) 作图子程序:chap7_4plot. m

```
close all;

figure(1);
plot(t,y(:,1),'r');
xlabel('time(s)');ylabel('state response');

figure(2);
plot(t,du,'r');
xlabel('time(s)');ylabel('derivative of control input');

figure(3);
plot(t,u,'r');
xlabel('time(s)');ylabel('control input');
```

7.3.5 新型反演自适应动态滑模控制

针对一阶系统式(7.22)、式(7.23)，定义 Lyapunov 函数

$$V_1 = \frac{1}{2}x^2 + \frac{1}{2}(\theta - \hat{\theta})^2 \tag{7.32}$$

其中 $\hat{\theta}$ 为未知项 θ 的估计值。

切换函数设计为

$$s = c_1 x + u + \hat{\theta}x^2 \tag{7.33}$$

其中 $c_1>0$。

可得

$$\dot{V}_1 = -c_1x^2 + xs + (\theta - \hat{\theta})(x^3 - \dot{\hat{\theta}})$$

由式(7.33)得

$$\dot{x} = u + \theta x^2 = -c_1 x + s - (\theta - \hat{\theta})x^2 \tag{7.34}$$

由于

$$\dot{s} = c_1\dot{x} + \dot{u} + \dot{\hat{\theta}}x^2 + 2\dot{x}\hat{\theta}x = v + (c_1 + 2\hat{\theta}x)\dot{x} + \dot{\hat{\theta}}x^2$$

将式(7.22)代入式(7.34)，得

$$\dot{s} = v + (c_1 + 2\hat{\theta}x)(u + \theta x^2) + \dot{\hat{\theta}}x^2 = v + (c_1 + 2\hat{\theta}x)u + \dot{\hat{\theta}}x^2 + (c_1 + 2\hat{\theta}x)\theta x^2$$

定义 Lyapunov 函数

$$V_2 = V_1 + \frac{1}{2}s^2$$

则

$$\begin{aligned}\dot{V}_2 = \dot{V}_1 + s\dot{s} &= -c_1x^2 + xs + (\theta - \hat{\theta})(\ \dot{\hat{\theta}} + x^3) + s[v + (c_1 + 2\hat{\theta}x)u + \dot{\hat{\theta}}x^2 + (c_1 + 2\hat{\theta}x)\theta x^2] \\ &= -c_1x^2 + xs + (\theta - \hat{\theta})(-\dot{\hat{\theta}} + x^3) + s[v + (c_1 + 2\hat{\theta}x)(u + \theta x^2) + \dot{\hat{\theta}}x^2]\end{aligned}$$

令动态控制律为

$$v = \dot{u} = -x - (c_1 + 2\hat{\theta}x)(u + \hat{\theta}x^2) - \dot{\hat{\theta}}x^2 - k\,\mathrm{sgn}(s) \tag{7.35}$$

则

$$\begin{aligned}V_2 &= -c_1x^2 + (\theta - \hat{\theta})(-\dot{\hat{\theta}} + x^3) + s[(c_1 + 2\hat{\theta}x)(\theta - \hat{\theta})x^2 - k\,\mathrm{sgn}(s)] \\ &= -c_1x^2 + (\theta - \hat{\theta})[s(c_1 + 2\hat{\theta}x)x^2 - \dot{\hat{\theta}} + x^3] - k\,|\,s\,|\end{aligned}$$

设计自适应律为

$$\dot{\hat{\theta}} = x^3 + sx^2(c_1 + 2\hat{\theta}x) \tag{7.36}$$

则

$$V_2 = -c_1x^2 - k\,|\,s\,| \leqslant 0$$

动态自适应滑模控制律由式(7.35)和式(7.36)组成。

7.3.6 仿真实例

针对一阶不确定性系统式(7.18)，取 $\theta=1$，系统初始状态为 $x(0)=0.5$。采用动态滑模

控制律式(7.35)，取 $c=2.0$ 的初始值取 0，仿真结果如图 7-16～图 7-18 所示。

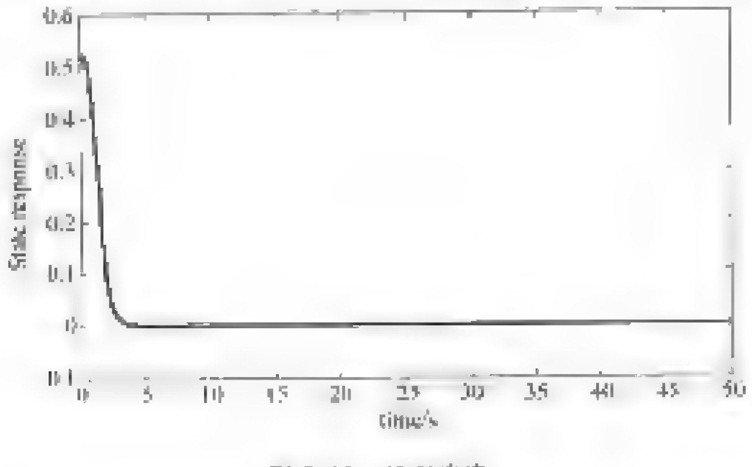


图 7-16　状态响应

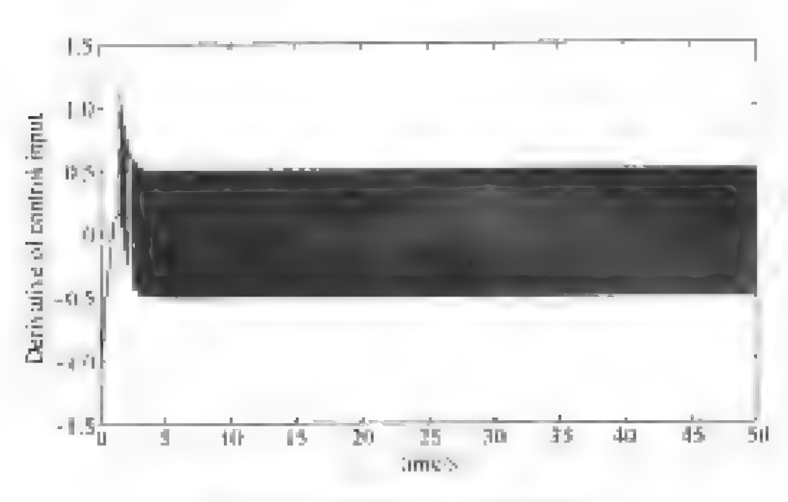


图 7-17　控制输入信号的导数 w

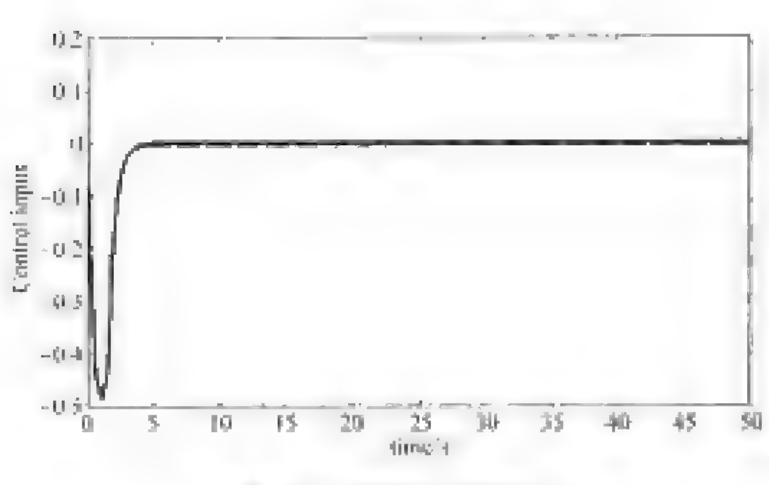


图 7-18　控制输入信号 u

仿真程序：

(1) 主程序(如图 7-19 所示)：chap7_5sim. mdl

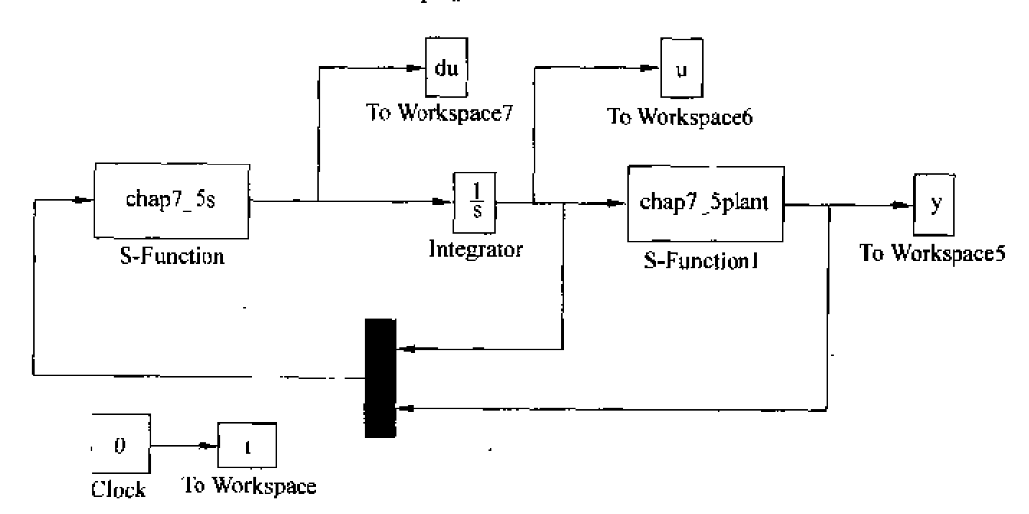


图 7-19 主程序图

(2) S 函数控制子程序：chap7_5s. m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
  case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
```

```
sys = simsizes(sizes);
x0 = [0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
c1 = 2;
s = c1 * u(2) + u(1) + x(1) * u(2)^2;
sys(1) = u(2)^3 + s * u(2)^2 * (c1 + 2 * x(1) * u(2));
function sys = mdlOutputs(t,x,u)
k = 0.5;
c1 = 2;
s = c1 * u(2) + u(1) + x(1) * u(2)^2;
ut = u(1);
dth = u(2)^3 + s * u(2)^2 * (c1 + 2 * x(1) * u(2));

sys(1) = - u(2) - (c1 + 2 * x(1) * u(2)) * (ut + x(1) * u(2)^2) - dth * u(2)^2 - k * sign(s);
```

(3) 被控对象子程序:chap7_5plant. m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 1;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 1;
sizes.NumInputs       = 1;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 0;

sys = simsizes(sizes);
```

```
x0=[0.5];
str=[];
ts=[];

function sys=mdlDerivatives(t,x,u)
th=1;
sys(1)=u+th*x(1)^2;
function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
```

(4) 作图子程序:chap7_5plot. m

```
close all;

figure(1);
plot(t,y(:,1),'r');
xlabel('time(s)');ylabel('state response');

figure(2);
plot(t,du,'r');
xlabel('time(s)');ylabel('derivative of control input');

figure(3);
plot(t,u,'r');
xlabel('time(s)');ylabel('control input');
```

参 考 文 献

1. Pieper J. First order dynamic sliding mode control. Proceedings of the 37th IEEE Conference on Decision & Control, Tampa, Florida USA, 1998. 2415~2420
2. Ramirez H S, Santiago O L. Adaptive dynamical sliding mode control via backstepping. Proceedings of the 32th Conference on Decision and Control, San Antonia, Texas, 1992, 1422~1427

第 8 章　基于干扰估计的滑模控制

8.1　一种简单滑模观测器的设计

8.1.1　系统描述

针对一阶系统

$$\dot{z} = z_0 + a(t) \tag{8.1}$$

其中 z 为待观测值，z_0 为已知，$a(t)$为不确定项。其上界为 $\bar{a}(t)$，即 $a(t)\leqslant\bar{a}(t)$。

定义观测误差为 $e=z-\hat{z}$，则针对一阶系统，切换函数设计为

$$s = e \tag{8.2}$$

对 z 构造如下滑模观测器：

$$\dot{\hat{z}} = z_0 + L\mathrm{sgn}(s) \tag{8.3}$$

其中 $L>\bar{a}(t)$。

s 的导数为

$$\dot{s} = \dot{z} - \dot{\hat{z}} = a(t) - L\mathrm{sgn}(s) \tag{8.4}$$

稳定性分析：

定义 Lyapunov 函数

$$V = \frac{1}{2}s^2$$

则

$$\dot{V} = s\dot{s} = s[a(t) - L\mathrm{sgn}(s)] = sa(t) - L\mid s\mid\leqslant a(t)\mid s\mid - L\mid s\mid = [a(t) - L]\mid s\mid\leqslant 0$$

说明滑模观测器式(8.3)满足滑模存在条件，在滑动模态下有 $z=\hat{z}$。

8.1.2　仿真实例

针对系统式(8.1)，取 $z_0=1.0$，$a(t)=3\sin(t)$，观测器参数取 $L=10$，系统初始状态为 0，观测器初始状态为 2。仿真结果如图 8-1 所示。可见，通过滑模观测器，可以估计出未知干扰，从而实现高精度控制。

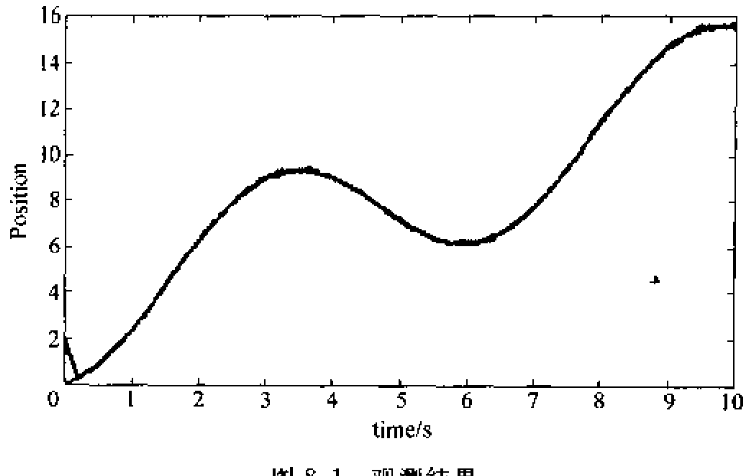


图 8-1 观测结果

仿真程序：

(1) 主程序：chap8_1.m

```
% Sliding Mode Observer
clear all;
close all;

xk = [0,2];

ts = 0.001;
T = 10;
TimeSet = [0:ts:T];
para = [];
[t,y] = ode45('chap8_1eq',TimeSet,xk,[],para);

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position');
```

(2) 子程序：chap8_1eq.m

```
function dx = PlantModel(t,x,flag,para)
dx = zeros(2,1);

at = 3 * sin(t);
z0 = 1.0;
L = 10;

dx(1) = z0 + at;
dx(2) = z0 + L * sign(x(1) - x(2));
```

8.2 基于干扰观测器的连续滑模控制

参考文献[1]，探讨通过设计干扰观测器来降低滑模切换项的增益，从而降低连续滑模控制的抖振。

8.2.1　系统描述

考虑如下伺服系统：

$$\ddot{\theta}=-b\dot{\theta}+u-f \tag{8.5}$$

其中 f 为外加干扰。

设位置指令信号为一常数 θ_{ref}，定义误差及其导数：

$$x_1=\theta_{\text{ref}}-\theta \tag{8.6}$$

$$x_2=\dot{x}_1=\dot{\theta}_{\text{ref}}-\dot{\theta}=-\dot{\theta},\dot{x}_2=\ddot{x}_1=\ddot{\theta}_{\text{ref}}-\ddot{\theta}=-\ddot{\theta} \tag{8.7}$$

将式(8.6)、式(8.7)代入式(8.5)，则得误差状态方程

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}0 & 1\\ 0 & -b\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}0\\ -1\end{bmatrix}u+\begin{bmatrix}0\\ 1\end{bmatrix}f \tag{8.8}$$

8.2.2　常规滑模控制器

滑模控制器设计为

$$u=u_s=\Psi x_1+K_f\text{sgn}(s) \tag{8.9}$$

其中

$$\Psi=\begin{cases}\alpha & \text{if} \quad sx_1>0\\ \beta & \text{if} \quad sx_1<0\end{cases} \tag{8.10}$$

$$b\geqslant c \tag{8.11}$$

$$\alpha>c(b-c) \tag{8.12}$$

$$\beta<c(b-c) \tag{8.13}$$

$$K_f>|f|_{\max} \tag{8.14}$$

切换函数为

$$s=cx_1+x_2 \tag{8.15}$$

稳定性分析：

$$\begin{aligned}\dot{s}&=c\dot{x}_1+\dot{x}_2=cx_2-bx_2-u+f=cx_2-bx_2-\Psi x_1-K_f\text{sgn}(s)+f\\&=(c-b)(s-cx_1)-\Psi x_1-K_f\text{sgn}(s)+f\\&=(c-b)s-c(c-b)x_1-\Psi x_1-K_f\text{sgn}(s)+f\\&=(c-b)s+[c(b-c)-\Psi]x_1+f-K_f\text{sgn}(s)\end{aligned}$$

则

$$\begin{aligned}s\dot{s}&=(c-b)s^2+[c(b-c)-\Psi]sx_1+fs-K_f|s|\\&\leqslant(c-b)s^2+[c(b-c)-\Psi]sx_1+(|f|_{\max}-K_f)|s|\end{aligned}$$

将式(8.10)~式(8.14)代入上式，得

$$s\dot{s}\leqslant 0 \tag{8.16}$$

由式(8.14)可知，当干扰 f 很大时，K_f 很大，此时会导致强的抖振。

8.2.3 带干扰观测器的滑模控制器

为了观测干扰 f，设计干扰观测器为

$$\begin{bmatrix}\dot{\hat{f}}\\ \dot{\hat{x}}_2\end{bmatrix}=\begin{bmatrix}0 & 0\\ 1 & -b\end{bmatrix}\begin{bmatrix}\hat{f}\\ \hat{x}_2\end{bmatrix}+\begin{bmatrix}0\\ -1\end{bmatrix}u+\begin{bmatrix}K_1\\ K_2\end{bmatrix}[x_2-\hat{x}_2] \tag{8.17}$$

其中 $\hat{f}$ 为对干扰 f 的估计，$\hat{x}_2$ 为对干扰 x_2 的估计，K_1 和 K_2 为通过极点配置的增益。

滑模控制器设计为

$$u_o=\hat{f} \tag{8.18}$$

$$u=u_s+u_o \tag{8.19}$$

其中 u_s 见式(8.9)。

采用干扰观测器前，根据式(8.8)和式(8.9)，有

$$\dot{x}_2=-bx_2-u_s+f \tag{8.20}$$

采用干扰观测器后，根据式(8.8)和式(8.19)，有

$$\dot{x}_2=-bx_2-u_s+f-\hat{f} \tag{8.21}$$

此时，干扰项 f 就变为 $f-\hat{f}$。可见，如果能够通过干扰观测器实现 $\hat{f}\to f$，则切换增益 K_f 大大降低，从而可有效地降低抖振。

稳定性分析：

定义 Lyapunov 函数为

$$V=V_1+V_2=\frac{1}{2}s^2+\frac{1}{2K_1}\tilde{f}^2+\frac{1}{2}\tilde{x}_2^2 \tag{8.22}$$

其中 $V_1=\frac{1}{2}s^2$，$V_2=\frac{1}{2K_1}\tilde{f}^2+\frac{1}{2}\tilde{x}_2^2$，$\tilde{f}=f-\hat{f}$，$\tilde{x}_2=x_2-\hat{x}_2$。则

$$\begin{aligned}\dot{s}&=c\dot{x}_1+\dot{x}_2=cx_2-bx_2-u+f=cx_2-bx_2-\Psi x_1-K_f\mathrm{sgn}(s)-\hat{f}+f\\&=(c-b)(s-cx_1)-\Psi x_1-K_f\mathrm{sgn}(s)+\tilde{f}\\&=(c-b)s-c(c-b)x_1-\Psi x_1-K_f\mathrm{sgn}(s)+\tilde{f}\\&=(c-b)s+[c(b-c)-\Psi]x_1+\tilde{f}-K_f\mathrm{sgn}(s)\end{aligned}$$

$$\begin{aligned}s\dot{s}=&(c-b)s^2+[c(b-c)-\Psi]sx_1+\tilde{f}s-K_f|s|\leqslant(c-b)s^2+\\&[c(b-c)-\Psi]sx_1+(|\tilde{f}|_{\max}-K_f)|s|\end{aligned}$$

即

$$\dot{V}_1\leqslant(c-b)s^2+[c(b-c)-\Psi]sx_1+(|\tilde{f}|_{\max}-K_f)|s|$$

$$\dot{V}_2=\frac{1}{K_1}\tilde{f}\dot{\tilde{f}}+\tilde{x}_2\dot{\tilde{x}}_2=\frac{1}{K_1}\tilde{f}(\dot{f}-\dot{\hat{f}})+\tilde{x}_2(\dot{x}_2-\dot{\hat{x}}_2) \tag{8.23}$$

由干扰观测器式(8.17)可知

$$\dot{\hat{f}}=K_1\tilde{x}_2 \tag{8.24}$$

$$\dot{\hat{x}}_2=\hat{f}-b\hat{x}_2-u+K_2\tilde{x}_2 \tag{8.25}$$

假设干扰为慢时变信号，即

$$\dot{f}=0 \tag{8.26}$$

将式(8.21)、式(8.24)～式(8.26)代入式(8.23)，得

$$\dot{V}_2=-\frac{1}{K_1}\tilde{f}\dot{\hat{f}}+\tilde{x}_2[-bx_2-u_s+f-\hat{f}-(\hat{f}-b\hat{x}_2-u+K_2\tilde{x}_2)]$$

$$
\begin{aligned}
&=-\frac{1}{K_1}\tilde{f}\dot{\hat{f}}+\tilde{x}_2(-bx_2-u_s+f-\hat{f}-\tilde{f}+b\hat{x}_2+u-K_2\tilde{x}_2)\\
&=-\tilde{f}\tilde{x}_2+\tilde{x}_2(-bx_2+f-\hat{f}+b\hat{x}_2-K_2\tilde{x}_2)\\
&=-\tilde{f}\tilde{x}_2-b\tilde{x}_2^2+\tilde{f}\tilde{x}_2-K_2\tilde{x}_2^2\\
&=-b\tilde{x}_2^2-K_2\tilde{x}_2^2=-(b+K_2)\tilde{x}_2^2
\end{aligned}
$$

当以下条件存在：

$$
\begin{gathered}
b \geqslant c \\
\Psi=\begin{cases}\geqslant c(b-c) & \text{if } sx_1 \geqslant 0 \\ \leqslant c(b-c) & \text{if } sx_1 \leqslant 0\end{cases} \\
K_f \geqslant |\tilde{f}|_{\max} \\
K_1>0 \\
K_2 \geqslant -b
\end{gathered}
\tag{8.27}
$$

则

$$
\dot{V}_1 \leqslant 0, \dot{V}_2 \leqslant 0, \dot{V} \leqslant 0 \tag{8.28}
$$

对比式(8.14)和式(8.27)可见，采用干扰观测器后，切换增益 K_f 值大大降低。有效地降低了抖振。

8.2.4 仿真实例

被控对象的误差状态方程为

$$
\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}0 & 1\\ 0 & -100\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}0\\ -1\end{bmatrix}u+\begin{bmatrix}0\\ 1\end{bmatrix}f
$$

系统的初始状态为$\begin{bmatrix}\frac{\pi}{6} & 0\end{bmatrix}$，设系统所受的外干扰为正弦干扰，$f=8.5+0.5\sin(2\pi t)$，则$|f|_{\max}=9.0$。取 $c=20, K_1=15000, K_2=50$。采用常规滑模控制器式(8.9)，取 $M=1$，$K_f=|f|_{\max}+0.001$，仿真结果如图 8-2～图 8-4 所示。采用带干扰观测器的滑模控制器式(8.19)，取 $M=2, K_f=0.1$，仿真结果如图 8-5～图 8-8 所示。

可见，采用干扰观测器，可降低切换项的增益，从而降低抖振。

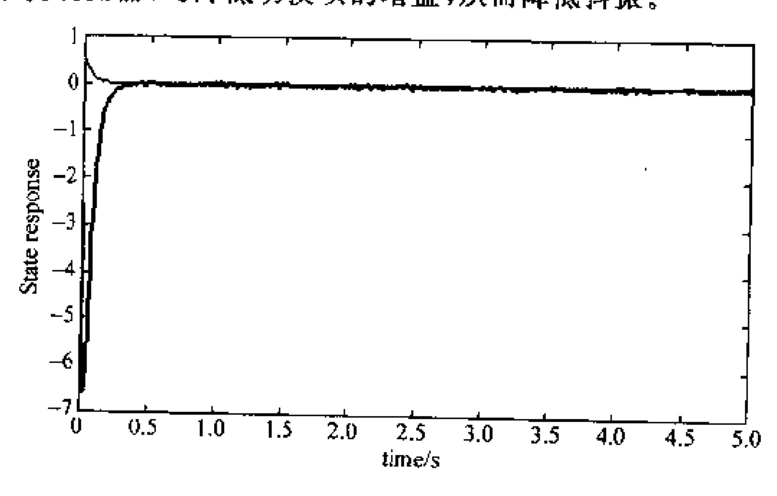


图 8-2　常规滑模控制误差状态响应（$M=1$）

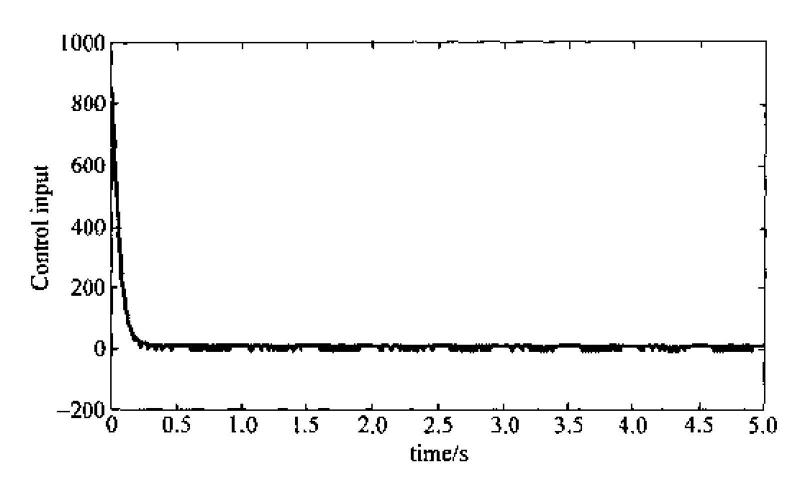


图 8-3 控制输入（$M=1$）

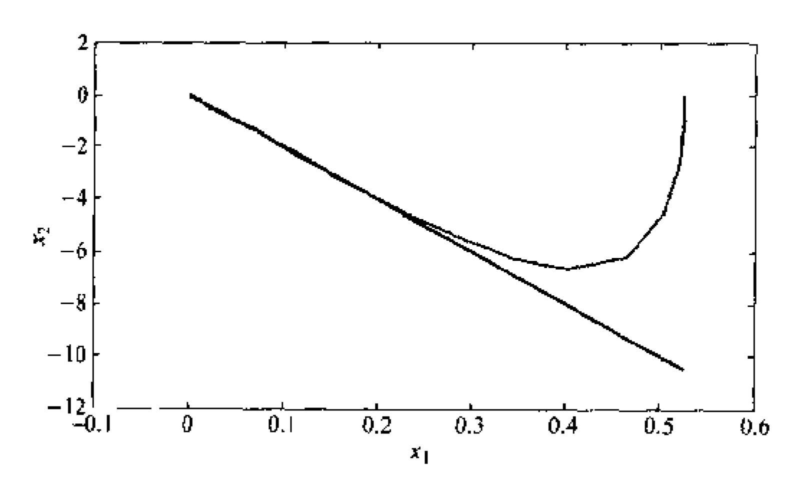


图 8-4 相轨迹（$M=1$）

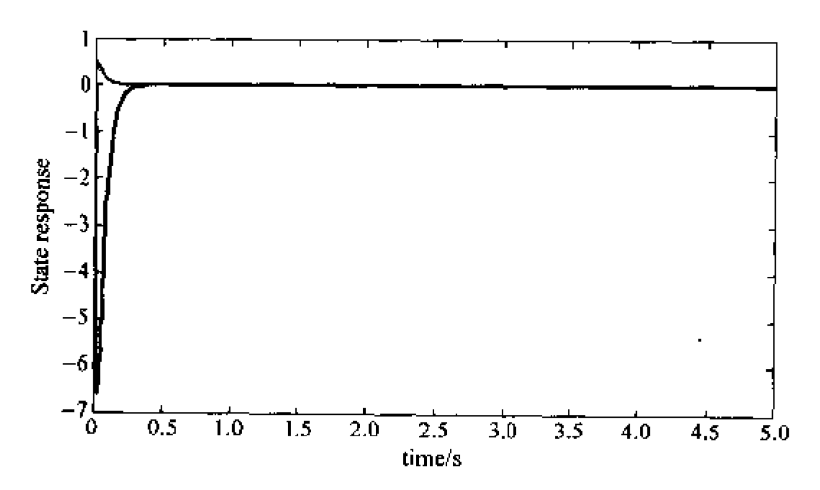


图 8-5 干扰补偿滑模控制误差状态响应（$M=2$）

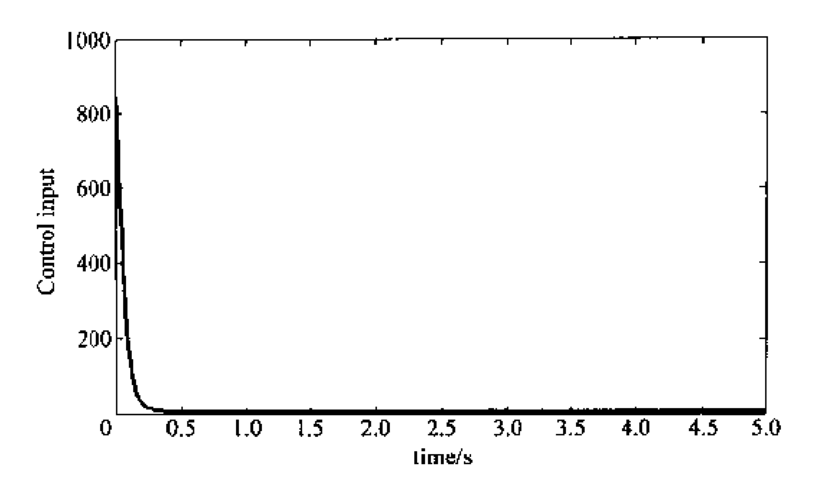


图 8-6　控制输入 ($M=2$)

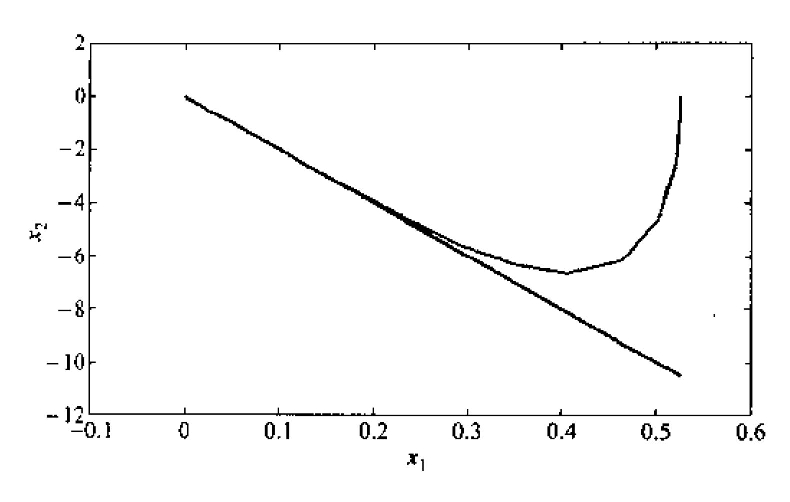


图 8-7　相轨迹($M=2$)

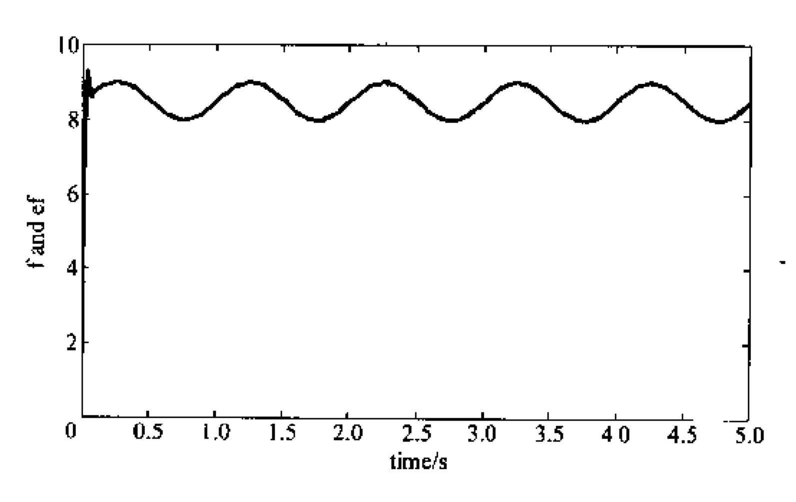


图 8-8　干扰观测结果($M=2$)

仿真程序：

(1) Simulink 主程序(如图 8-9 所示)：chap8_2sim.mdl

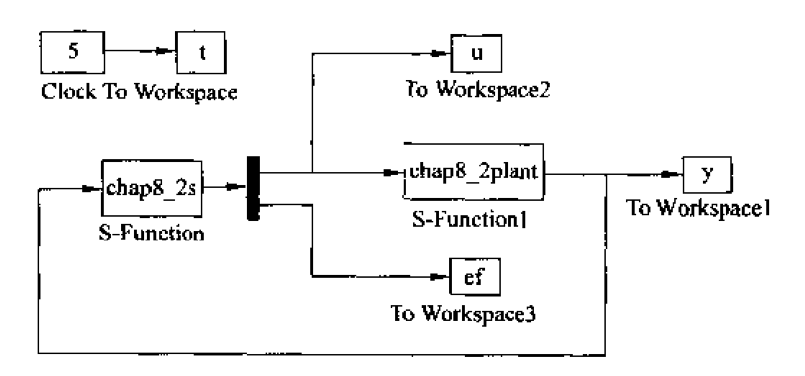


图 8-9 主程序图

(2) 控制器 S 函数:chap8_2s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [];

function sys=mdlOutputs(t,x,u)
c=20;
b=100;
z=c*(b-c);

alfa=z+10;      %alfa>z
beta=z-10;      %beta<z

x1=u(1);
x2=u(2);
```

```
x3 = u(3);

s = c * x1 + x2;
fmax = 9;

if s * x1>0
   fai = alfa;
elseif s * x1<0
   fai = beta;
end

uo = x3;

M = 2;
if M == 1
     Kf = fmax + 0.001;
     us = fai * x1 + Kf * sign(s);
   ut = us;
elseif M == 2
     Kf = 0.1;
     us = fai * x1 + Kf * sign(s);
   ut = us + uo;
end

sys(1) = ut;
sys(2) = uo;
```

(3) 被控对象 S 函数:chap8_2plant.m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
```

```
sizes = simsizes;
sizes.NumContStates  = 4;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 4;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [pi/6,0,0,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
b = 100;
f = 8.5 + 0.5 * sin(2 * pi * t);

K1 = 15000;
K2 = 50;

sys(1) = x(2);
sys(2) = - b * x(2) - u + f;
sys(3) = K1 * (x(2) - x(4));
sys(4) = x(3) - b * x(4) - u + K2 * (x(2) - x(4));

function sys = mdlOutputs(t,x,u)
f = 8.5 + 0.5 * sin(2 * pi * t);

sys(1) = x(1);
sys(2) = x(2);
sys(3) = x(3);
sys(4) - f;
```

(4) 作图程序:chap8_2plot.m

```
close all;
c = 20;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('State response');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('control input');

figure(3);
plot(y(:,1),y(:,2),'r',y(:,1), - c * y(:,1),'b');
xlabel('x1');ylabel('x2');

figure(4);
```

```
plot(t,y(:,4),'r',t,ef(:,1),'b');
xlabel('time(s)');ylabel('f and ef');
```

8.3　基于干扰观测器的离散滑模控制

参考文献[2]，探讨基于干扰观测器的离散滑模控制设计方法，从而降低离散滑模控制的抖振。

8.3.1　系统描述

考虑如下线性参数不确定系统：

$$\boldsymbol{x}(k+1)=(\boldsymbol{A}+\Delta\boldsymbol{A})\boldsymbol{x}(k)+\boldsymbol{b}u(k)+\boldsymbol{f}(k) \tag{8.29}$$

其中 $\boldsymbol{b}>0$。

假设系统满足匹配条件：

$$\Delta\boldsymbol{A}=\boldsymbol{b}\tilde{\boldsymbol{A}},\ \boldsymbol{f}=\boldsymbol{b}\tilde{f} \tag{8.30}$$

则系统式(8.29)可写为

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{b}[u(k)+d(k)] \tag{8.31}$$

其中 $d(k)=\tilde{\boldsymbol{A}}\boldsymbol{x}(k)+\tilde{f}(k)$。

8.3.2　基于干扰观测器的离散滑模控制器

滑模控制器及干扰观测器设计为

$$u(k)=-\hat{d}(k)+(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b})^{-1}[\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}_{\mathrm{r}}(k+1)-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}(k)+qs(k)-\eta\operatorname{sgn}(s(k))] \tag{8.32}$$

$$\hat{d}(k)=\hat{d}(k-1)+(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b})^{-1}g[s(k)-qs(k-1)+\eta\operatorname{sgn}(s(k-1))] \tag{8.33}$$

其中 $\tilde{d}(k)=d(k)-\hat{d}(k)$，$\eta$，$q$，$g$ 为正的常数。

由滑模控制器和干扰观测器可以得到以下两条推论：

推论 1　滑模及干扰估计误差的动态特性满足

$$s(k+1)=qs(k)-\eta\operatorname{sgn}(s(k))+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\tilde{d}(k) \tag{8.34}$$

$$\tilde{d}(k+1)=(1-g)\tilde{d}(k)+d(k+1)-d(k) \tag{8.35}$$

证明　将式(8.31)和式(8.32)代入下式中，得

$$\begin{aligned}s(k+1)&=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{e}(k+1)=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}(k+1)-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}_{\mathrm{r}}(k+1)\\&=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}u(k)+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}d(k)-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}_{\mathrm{r}}(k+1)\\&=qs(k)-\eta\operatorname{sgn}(s(k))+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\tilde{d}(k)\end{aligned}$$

将式(8.33)和式(8.34)代入下式中，得

$$\begin{aligned}\tilde{d}(k+1)&=d(k+1)-\hat{d}(k+1)\\&=d(k+1)-\hat{d}(k)-(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b})^{-1}g[s(k+1)-qs(k)+\eta\operatorname{sgn}(s(k))]\\&=d(k+1)-d(k)+\tilde{d}(k)-(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b})^{-1}g(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b})\tilde{d}(k)\\&=d(k+1)-d(k)+(1-g)\tilde{d}(k)\end{aligned}$$

推论 2 存在某一正常数 m，如果 $|d(k+1)-d(k)|<m$，则存在 k_0，当 $k>k_0$ 时，$\tilde{d}(k)<m/g$，其中 $0<g<1$。

证明 首先将 $\tilde{d}(k)$ 分解为

$$\tilde{d}(k)=\tilde{d}_1(k)+\tilde{d}_2(k) \tag{8.36}$$

令 $\tilde{d}_1(0)=0$，则

$$\tilde{d}_2(0)=\tilde{d}(0)$$

由于

$$\tilde{d}(k+1)=\tilde{d}_1(k+1)+\tilde{d}_2(k+1)$$

令

$$\tilde{d}_1(k+1)=(1-g)\tilde{d}_1(k)+d(k+1)-d(k) \tag{8.37}$$

则

$$\tilde{d}_2(k+1)-(1-g)\tilde{d}_2(k) \tag{8.38}$$

采用归纳法证明，首先证明 $\tilde{d}_1(k)<m/g$。

(1) 当 $k=0$ 时，由已知得：$\tilde{d}_1(0)=0<m/g$。

(2) 假设 $|\tilde{d}_1(k)|<m/g$，则由式(8.37)及 $0<g<1$ 得

$$|\tilde{d}_1(k+1)|\leqslant(1-g)|\tilde{d}_1(k)|+|d(k+1)-d(k)|<(1-g)\frac{m}{g}+m=\frac{m}{g}$$

由上两式得

$$|\tilde{d}_1(k)|<m/g, k\geqslant 0 \tag{8.39}$$

由式(8.38)及 $0<1-g<1$ 得

$$\tilde{d}_2(k+1)=(1-g)\tilde{d}_2(k)\leqslant(1-g)|\tilde{d}_2(k)|<|\tilde{d}_2(k)|$$

因此，$\tilde{d}_2(k)$ 逐渐递减。则存在 k_0'，当 $k>k_0'$ 时，$\tilde{d}_2(k)$ 可任意小。

由上述分析可知，存在 k_0，当 $k>k_0$ 时，有

$$|\tilde{d}(k)|=|\tilde{d}_1(k)+\tilde{d}_2(k)|\leqslant|\tilde{d}_1(k)|+|\tilde{d}_2(k)|<\frac{m}{g}$$

8.3.3 稳定性分析

假设如下条件满足，则控制器式(8.32)稳定：

(1) $0<q<1, 0<g<1$；

(2) 存在某一正常数 m，$|d(k+1)-d(k)|<m$；

(3) $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}(m/g)<\eta$。

证明 令 $v(k)=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\tilde{d}(k)$，则

$$|v(k)|<\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}<\eta, \text{即} -\eta<v(k)<\eta, -\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}<v(k)<\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}$$

$$s(k+1)=qs(k)-\eta\mathrm{sgn}(s(k))+v(k)$$

分以下四种情况进行讨论。

(1) 当 $s(k)\geqslant\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\dfrac{m}{g}+\eta>0$ 时：

$$s(k+1)-s(k)=(q-1)s(k)-\eta+v(k)<0$$

$$s(k+1)+s(k)=(q+1)s(k)-\eta+v(k)\geqslant(q+1)\left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta\right)-\eta+v(k)$$

$$=q\left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta\right)+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+v(k)>0$$

则

$$s(k+1)^2<s(k)^2$$

(2) 当 $s(k)\leqslant-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}-\eta\leqslant 0$ 时：

$$s(k+1)-s(k)=(q-1)s(k)+\eta+v(k)>0$$

$$s(k+1)+s(k)=(q+1)s(k)+\eta+v(k)<s(k)+\eta+v(k)$$

$$\leqslant-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}-\eta+\eta+v(k)=-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+v(k)<0$$

则

$$s(k+1)^2<s(k)^2$$

(3) 当 $0<s(k)<\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta$ 时：

$$s(k+1)=qs(k)-\eta+v(k)<q\left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta\right)-\eta+v(k)$$

$$<q\left(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta\right)<\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta$$

$$s(k+1)=qs(k)-\eta+v(k)>-\eta+v(k)>-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}-\eta$$

则

$$|s(k+1)|<\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta$$

(4) 当 $-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}-\eta<s(k)<0$ 时：

$$s(k+1)=qs(k)+\eta+v(k)>s(k)+\eta+v(k)$$

$$>-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}-\eta+\eta+v(k)>-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}-\eta$$

$$s(k+1)=qs(k)+\eta+v(k)<\eta+v(k)<\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta$$

则

$$|s(k+1)|<\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta$$

通过上述分析，可得到以下结论：

当 $|s(k)|\geqslant\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta$ 时，满足离散滑模到达条件：

$$s(k+1)^2<s(k)^2 \tag{8.40}$$

当 $|s(k)|<\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta$ 时：可见，只要使 $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta$ 足够小，$s(k)$ 将趋于零。

$$|s(k+1)|<\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\frac{m}{g}+\eta \tag{8.41}$$

8.3.4 仿真实例

被控对象为

$$G(s)=\frac{133}{s^2+25s}$$

采样时间取 0.001s,其离散化方程为

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{b}[u(k)+d(k)]$$

其中 $\boldsymbol{A}=\begin{bmatrix}1 & 0.001\\0 & 0.9753\end{bmatrix}$,$\boldsymbol{b}=\begin{bmatrix}0.0001\\0.1314\end{bmatrix}$,$d(k)$为外加干扰。

设所受干扰为正弦干扰,$d(k)=1.5\sin(4\pi t)$。指令为正弦信号 $x_r(k)=0.5\sin(4\pi t)$。控制器参数取 $\boldsymbol{c}^{\mathrm{T}}=[15\quad 1]$,$q=0.80$,$g=0.95$,$m=0.01$,$\eta$ 取 $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}(m/g)+0.001$。系统初始状态为$[1.5\quad 0]$。仿真结果如图 8-10～图 8-13 所示。

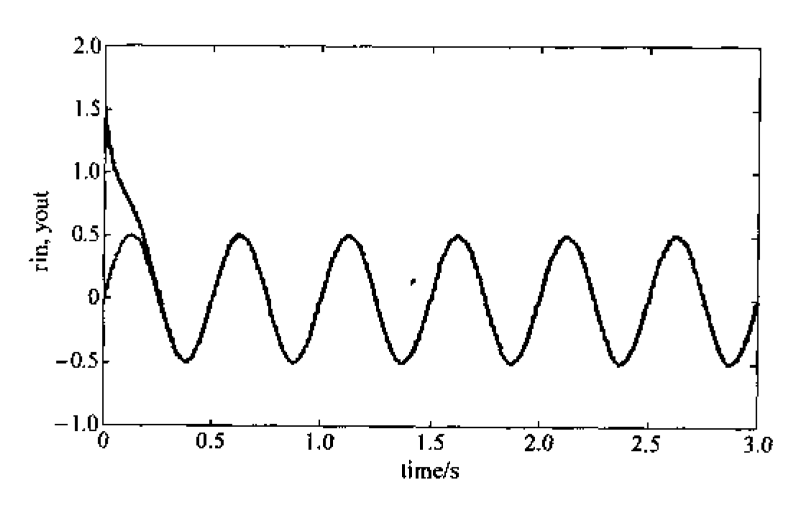


图 8-10 正弦跟踪

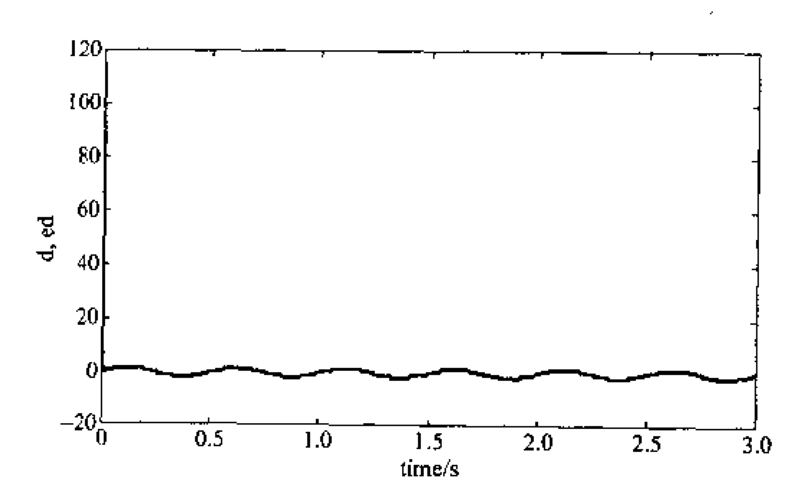


图 8-11 干扰观测结果

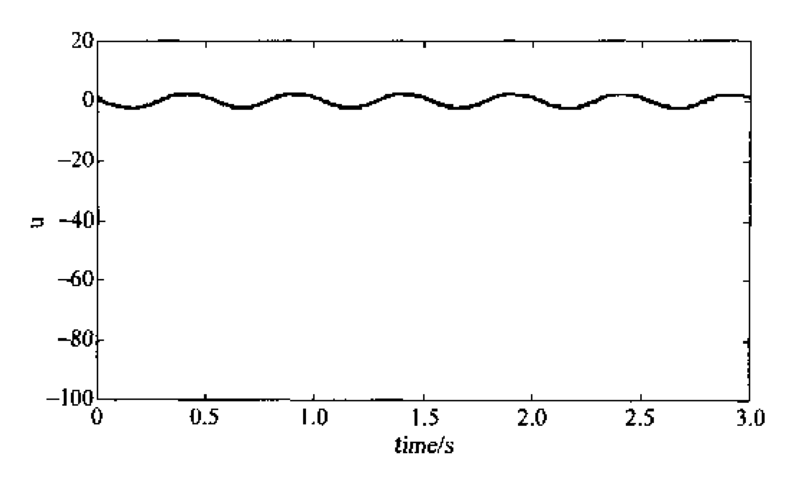


图 8-12　控制输入

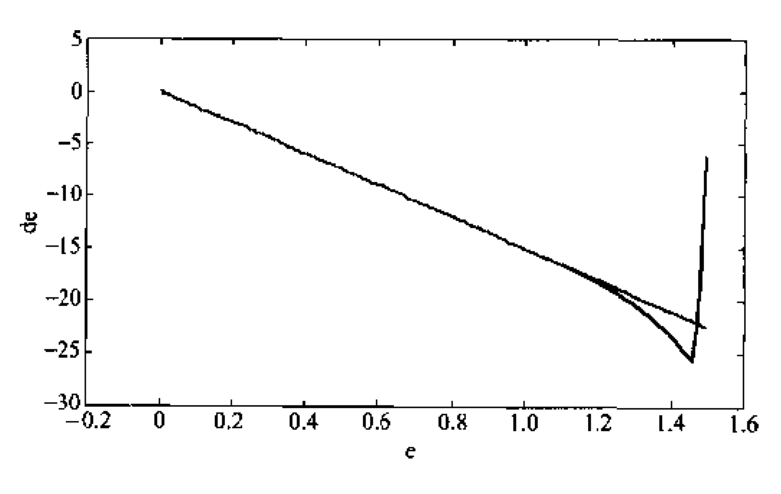


图 8-13　相轨迹

仿真程序：chap8_3. m

```
% VSS controller based on decoupled disturbance compensator
clear all;
close all;

ts = 0.001;
a = 25;
b = 133;
sys = tf(b,[1,a,0]);
dsys = c2d(sys,ts,'z');
[num,den] = tfdata(dsys,'v');

A = [0,1;0,-a];
B = [0;b];
C = [1,0];
D = 0;
% Change transfer function to discrete position equation
[A1,B1,C1,D1] = c2dm(A,B,C,D,ts,'z');
```

```
A = A1;
b = B1;
Ce = [15,1];
q = 0.80;                    % 0<q<1
g = 0.95;

m = 0.010;                   % m>abs(d(k+1) - d(k))

eq = Ce * b * m/g + 0.0010;    % eq>abs(Ce * b * m/g);0<eq/fai<q<1

x_1 = [1.5;0];
s_1 = 0;
u_1 = 0;
d_1 = 0;ed_1 = 0;
r_1 = 0;r_2 = 0;dr_1 = 0;

for k = 1:1:3000
time(k) = k * ts;

d(k) = 1.5 * sin(2 * 2 * pi * k * ts);
d_1 = d(k);

x = A * x_1 + b * (u_1 + d(k));

r(k) = 0.50 * sin(2 * 2 * pi * (k) * ts);
% Using Waitui method
   dr(k) = (r(k) - r_1)/ts;
   dr_1 = (r_1 - r_2)/ts;
   r1(k) = 2 * r(k) - r_1;
   dr1(k) = 2 * dr(k) - dr_1;

   xr = [r(k);dr(k)];
   xr1 = [r1(k);dr1(k)];

     e(k) = x(1) - xr(1);
     de(k) = x(2) - xr(2);
     s(k) = Ce * (x - xr);

   ed(k) = ed_1 + inv(Ce * b) * g * (s(k) - q * s_1 + eq * sign(s_1));

   u(k) = - ed(k) + inv(Ce * b) * (Ce * xr1 - Ce * A * x + q * s(k) - eq * sign(s(k)));

   r_2 = r_1;r_1 = r(k);
   dr_1 = dr(k);

   ed_1 = ed(k);
    x_1 = x;
    s_1 = s(k);
```

```
    x1(k) = x(1);
    x2(k) = x(2);
    u_1 = u(k);
end
figure(1);
plot(time,r,'r',time,x1);
xlabel('time(s)');ylabel('rin,yout');
figure(2);
plot(time,d,'r',time,ed,'b');
xlabel('time(s)');ylabel('d,ed');
figure(3);
plot(time,u,'r');
xlabel('time(s)');ylabel('u');
figure(4);
plot(e,de,'b',e,-Ce(1)*e,'r');
xlabel('e');ylabel('de');
```

8.3.5 基于饱和函数的控制器

为了进一步降低抖振，采用饱和函数的方法。则滑模控制器式(8.32)及干扰观测器式(8.33)变为

$$u(k)=-\hat{d}(k)+(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b})^{-1}\left[\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}_{\mathrm{r}}(k+1)-\boldsymbol{c}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}(k)+qs(k)-\eta\operatorname{sat}\left(\frac{s(k)}{\phi}\right)\right] \tag{8.42}$$

$$\hat{d}(k)=\hat{d}(k-1)+(\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b})^{-1}g\left[s(k)-qs(k-1)+\eta\operatorname{sat}\left(\frac{s(k-1)}{\phi}\right)\right] \tag{8.43}$$

其中 $\tilde{d}(k)=d(k)-\hat{d}(k)$，$\eta,q,g$ 为正的常数。

饱和函数为

$$\operatorname{sat}\left(\frac{s(k)}{\phi}\right)=\begin{cases}1 & s(k)\geqslant\phi\\ \dfrac{s(k)}{\phi} & |s(k)|<\phi\\ -1 & s(k)\leqslant-\phi\end{cases} \tag{8.44}$$

可得到以下两条推论。

推论 1　类似式(8.34)、式(8.35)，滑模及干扰估计误差的动态特性满足：

$$s(k+1)=qs(k)-\eta\operatorname{sat}\left(\frac{s(k)}{\phi}\right)+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\tilde{d}(k) \tag{8.45}$$

$$\tilde{d}(k+1)=(1-g)\tilde{d}(k)+d(k+1)-d(k) \tag{8.46}$$

推论 2　当 $\eta>\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\dfrac{m}{g}$ 且 $\eta>|\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\tilde{d}(k)|$，$0<q<1$，$\dfrac{\eta}{\phi}<q$ 时，$|s(k)|<\phi$，其中 ϕ 为小于 1 的正常数。

推论 2 的证明　分以下三种情况进行讨论。

(1) 当 $s(k)\geqslant\phi$ 时，由于 $\eta>\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\dfrac{m}{g}>\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\tilde{d}(k)$，则

$$s(k+1)=qs(k)-\eta+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{b}\tilde{d}(k)<qs(k)<s(k)$$

$$s(k+1)=qs(k)-\eta+c^{\mathrm{T}}b\tilde{d}(k)\geqslant q\phi-\eta+c^{\mathrm{T}}b\tilde{d}(k)>c^{\mathrm{T}}b\tilde{d}(k)>-\eta>-q\phi>-\phi$$

即
$$s(k+1)>-\phi。$$

结论：当 $s(k)\geqslant\phi$ 时，$s(k+1)$ 将逐渐减小，直至进入边界层。

(2) 当 $s(k)\leqslant-\phi$ 时：

$$s(k+1)=qs(k)+\eta+c^{\mathrm{T}}b\tilde{d}(k)>qs(k)>s(k)$$
$$s(k+1)=qs(k)+\eta+c^{\mathrm{T}}b\tilde{d}(k)\leqslant-q\phi+\eta+c^{\mathrm{T}}b\tilde{d}(k)$$

由于 $c^{\mathrm{T}}b\tilde{d}(k)<\eta<\phi$，则

$$s(k+1)<-q\phi+\eta+\phi$$

即
$$s(k+1)<\phi$$

结论：当 $s(k)\leqslant-\phi$ 时，$s(k+1)$ 将逐渐减小，直至进入边界层。

(3) 当 $|s(k)|<\phi$ 时：

$$s(k+1)=qs(k)-\eta\frac{s(k)}{\phi}+c^{\mathrm{T}}b\tilde{d}(k)=\left(q-\frac{\eta}{\phi}\right)s(k)+c^{\mathrm{T}}b\tilde{d}(k)$$

分两步讨论：由于 $q-\frac{\eta}{\phi}>0$，则当 $0\leqslant s(k)<\phi$ 时：

$$s(k+1)<\left(q-\frac{\eta}{\phi}\right)\phi+c^{\mathrm{T}}b\tilde{d}(k)=q\phi-\eta+c^{\mathrm{T}}b\tilde{d}(k)<q\phi<\phi$$

当 $-\phi<s(k)<0$ 时：

$$s(k+1)>\left(q-\frac{\eta}{\phi}\right)(-\phi)+c^{\mathrm{T}}b\tilde{d}(k)=-q\phi+\eta+c^{\mathrm{T}}b\tilde{d}(k)>-q\phi>-\phi$$

结论：当 $|s(k)|<\phi$ 时，$s(k+1)$ 将保持在边界层之内。

通过上述分析可知，当 $k>k_0$ 时，$s(k)$ 将进入边界层，并保持在边界层之内。

8.3.6 稳定性分析

假设如下条件满足，则控制器式(8.42)稳定：

(1) $0<q<1$，$0<g<1$，$q\phi>\eta$；

(2) 存在某一正常数 m，满足 $|d(k+1)-d(k)|<m$；

(3) $\eta>c^{\mathrm{T}}b\frac{m}{g}$ 且 $\eta>|c^{\mathrm{T}}b\tilde{d}(k)|$。

证明 由推论 2 可知，当 $|s(k)|<\phi$ 时，$s(k+1)$ 将保持在边界层之内。

当 $|s(k)|\geqslant\phi$ 时，分两种情况讨论离散滑模到达条件：

(1) 当 $s(k)\geqslant\phi$ 时：

$$s(k+1)=qs(k)-\eta+c^{\mathrm{T}}b\tilde{d}(k)<qs(k)<s(k)$$

即

$$s(k+1)-s(k)<0$$

由于 $q\phi>\eta$，$\phi-\eta>q\phi-\eta>0$，则

$$s(k+1)+s(k)=(q+1)s(k)-\eta+c^{\mathrm{T}}b\tilde{d}(k)\geqslant(q+1)\phi-\eta+c^{\mathrm{T}}b\tilde{d}(k)$$
$$=q\phi+\phi-\eta+c^{\mathrm{T}}b\tilde{d}(k)>\eta+c^{\mathrm{T}}b\tilde{d}(k)>0$$

即

$$s(k+1)+s(k)>0$$

则

$$s(k+1)^2<s(k)^2$$

（2）当 $s(k)\leqslant -\phi$ 时：

$$s(k+1)=qs(k)+\eta+c^{\mathrm{T}}b\tilde{d}(k)>qs(k)>s(k)$$

即

$$s(k+1)-s(k)>0$$

$$\begin{aligned}s(k+1)+s(k)&=(q+1)s(k)+\eta+c^{\mathrm{T}}b\tilde{d}(k)\leqslant -(q+1)\phi+\eta+c^{\mathrm{T}}b\tilde{d}(k)\\&=-q\phi-\phi+\eta+c^{\mathrm{T}}b\tilde{d}(k)=-q\phi+\eta-\phi+c^{\mathrm{T}}b\tilde{d}(k)<-\phi+c^{\mathrm{T}}b\tilde{d}(k)\\&<-\eta+c^{\mathrm{T}}b\tilde{d}(k)<0\end{aligned}$$

即

$$s(k+1)+s(k)<0$$

则

$$s(k+1)^2<s(k)^2$$

8.3.7　仿真实例

被控对象为

$$G(s)=\frac{133}{s^2+25s}$$

采样时间取 0.001s，其离散化方程为

$$x(k+1)=Ax(k)+b[u(k)+d(k)]$$

其中 $A=\begin{bmatrix}1 & 0.001\\0 & 0.9753\end{bmatrix}$，$b=\begin{bmatrix}0.0001\\0.1314\end{bmatrix}$，$d(k)$ 为外加干扰。

设所受干扰为正弦干扰，$d(k)=1.5\sin(4\pi t)$。指令为正弦信号 $x_r(k)=0.5\sin(4\pi t)$。控制器参数取 $c^{\mathrm{T}}=[15\quad 1]$，$q=0.80$，$\phi=0.05$，$g=0.95$，$m=0.01$。系统初始状态为[1.5　0]。仿真结果如图 8-14～图 8-17 所示。

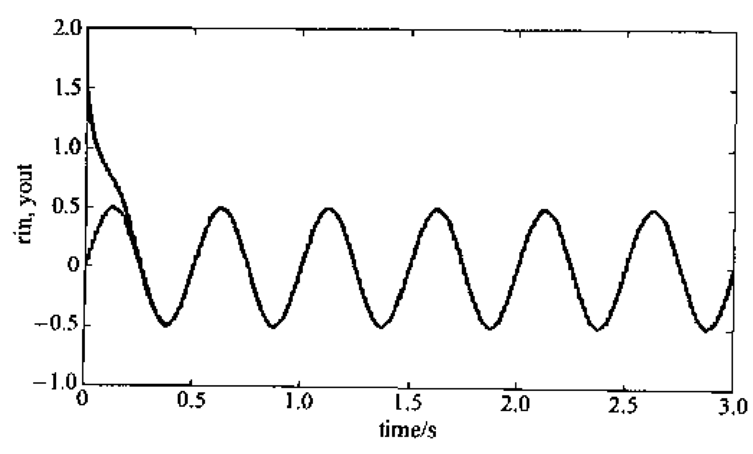


图 8-14　正弦跟踪

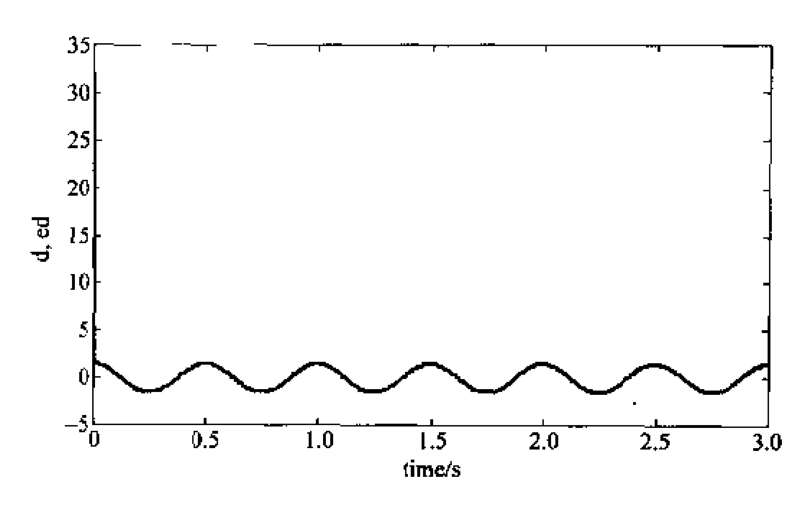


图 8-15 干扰观测结果

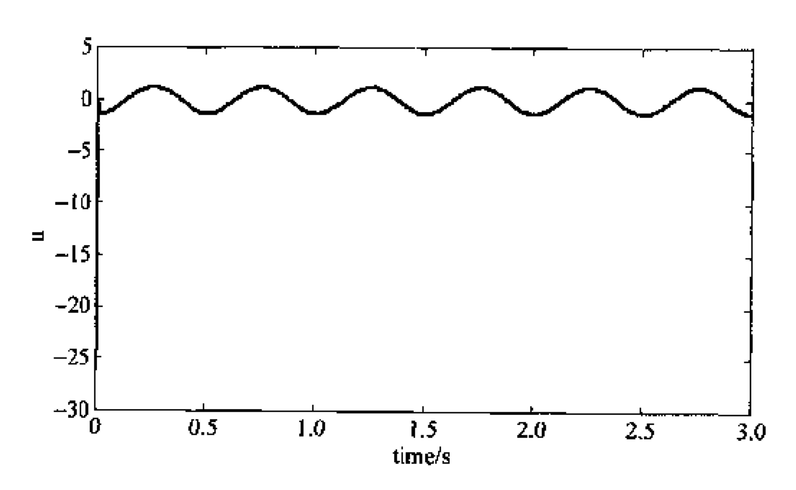


图 8-16 控制输入

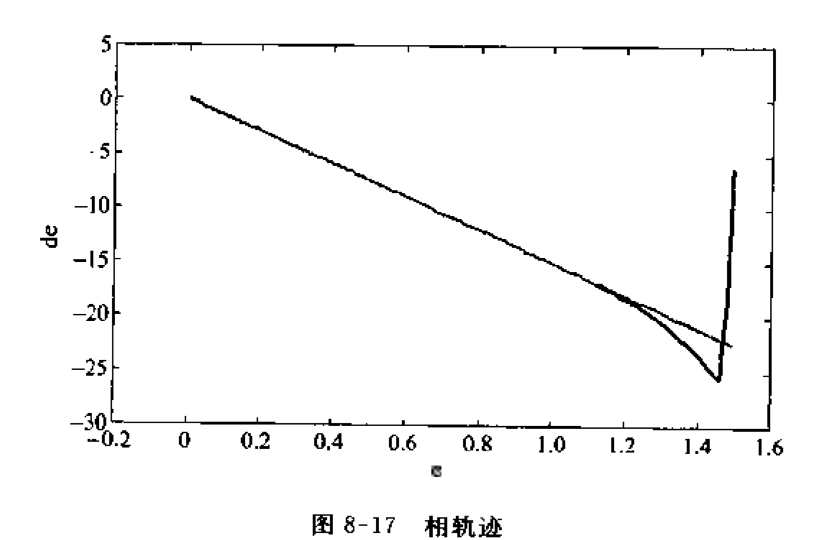


图 8-17 相轨迹

仿真程序：chap8_4. m

```
% VSS controller based on decoupled disturbance compensator
```

```
clear all;
close all;

ts = 0.001;
a = 20;b = 500;
sys = tf(b,[1,a,0]);
dsys = c2d(sys,ts,'z');
[num,den] = tfdata(dsys,'v');

A = [0,1;0, - a];
B = [0;b];
C = [1,0];
D = 0;
% Change transfer function to continous position equation
[A1,B1,C1,D1] = c2dm(A,B,C,D,ts,'z');
A = A1;
b = B1;
Ce = [15,1];
q = 0.80;                  % 0<q<1
g = 0.95;

m = 0.010;                 % m>abs(d(k + 1) - d(k))

eq = Ce * b * m/g + 0.0010;   % eq>abs(Ce * b * m/g);0<eq/fai<q<1
fai = 0.05;

x_1 = [1.5;0];
s_1 = 0;
u_1 = 0;
d_1 = 0;ed_1 = 0;
r_1 = 0;r_2 = 0;dr_1 = 0;

for k = 1:1:3000
time(k) = k * ts;

d(k) = 1.5 * sin(2 * 2 * pi * k * ts);
d_1 = d(k);

x = A * x_1 + b * (u_1 + d(k));

r(k) = 0.50 * sin(2 * 2 * pi * (k) * ts);
% Using Waitui method
   dr(k) = (r(k) - r_1)/ts;
   dr_1 = (r_1 - r_2)/ts;
   r1(k) = 2 * r(k) - r_1;
   dr1(k) = 2 * dr(k) - dr_1;

   xr = [r(k);dr(k)];
   xr1 = [r1(k);dr1(k)];

    e(k) = x(1) - xr(1);
```

```
    de(k) = x(2) - xr(2);
    s(k) = Ce * (x - xr);

 % Saturated function
 if s(k)/fai>1
    sat(k) = 1;
 elseif abs(s(k)/fai)<= 1
    sat(k) = s(k)/fai;
 elseif s(k)/fai< - 1
    sat(k) = - 1;
 end

 if s_1/fai>1
    sat_1 = 1;
 elseif abs(s_1/fai)<= 1
    sat_1 = s_1/fai;
 elseif s_1/fai< - 1
    sat_1 = - 1;
 end

    ed(k) = ed_1 + inv(Ce * b) * g * (s(k) - q * s_1 + eq * sat_1);
    u(k) = - ed(k) + inv(Ce * b) * (Ce * xr1 - Ce * A * x + q * s(k) - eq * sat(k));

   r_2 = r_1;r_1 = r(k);
   dr_1 = dr(k);

   ed_1 = ed(k);
    x_1 = x;
    s_1 = s(k);

    x1(k) = x(1);
    x2(k) = x(2);
    u_1 = u(k);
end
figure(1);
plot(time,r,'r',time,x1);
xlabel('time(s)');ylabel('rin,yout');
figure(2);
plot(time,d,'r',time,ed,'b');
xlabel('time(s)');ylabel('d,ed');
figure(3);
plot(time,u,'r');
xlabel('time(s)');ylabel('u');
figure(4);
plot(e,de,'b',e, - Ce(1) * e,'r');
xlabel('e');ylabel('de');
```

8.4　灰色滑模控制

系统被噪声污染后，原始数列呈现出离乱的情况。离乱的数列即灰色数列，或者灰色过程，对灰色过程建立的模型，称为灰色模型。利用灰色模型进行参数估计，称为灰色预测。

利用灰色预测设计滑模控制器，称为灰色滑模控制 [3~5]。

8.4.1　系统描述

假如 $D(x,k)$ 是加在 $u(k)$ 上的干扰，设采样时间为 T。

二阶离散位置状态方程为

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}u(k)+\boldsymbol{B}D(x,k) \tag{8.47}$$

其中 $\boldsymbol{x}(k)=\begin{bmatrix}x_1(k)\\x_2(k)\end{bmatrix}$，$x_1(k)$ 代表实际位置，$x_2(k)$ 代表位置变化率。

对于位置跟踪，设理想输入信号为 $r(k)$，则系统误差和误差变化率分别为

$$e(k)=r(k)-x_1(k)$$

$$\mathrm{de}(k)=\mathrm{dr}(k)-x_2(k)$$

设 $\boldsymbol{R}(k)=\begin{bmatrix}r(k)\\\mathrm{dr}(k)\end{bmatrix}$，$D(x,k)$ 代表系统的不确定部分，其中包括参数不确定部分和外部干扰。$D(x,k)$ 可表示为

$$D(x,k)=V_1x_1(k)+V_2x_2(k)+\cdots+V_nx_n(k)+d(k) \tag{8.48}$$

通过建立 $D(x,k)$ 的原始数列，并通过灰色预测，便可实现对干扰参数 V_i 和 $d(k)$ 的精确估计。

8.4.2　灰色滑模控制器的设计

1. 滑模控制器的设计

滑模面定义为

$$s=\boldsymbol{C}[\boldsymbol{R}(k)-\boldsymbol{x}(k)]=0 \tag{8.49}$$

其中向量 $\boldsymbol{C}$ 符合滑模的稳定条件，$\boldsymbol{C}=[C_1C_2\cdots C_n]$，$C_n=1$。

设

$$\mathrm{d}s(k)=s(k+1)-s(k) \tag{8.50}$$

根据指数趋近律：

$$\mathrm{d}s(k)=-\varepsilon T\mathrm{sgn}(s(k))-qTs(k),\ \varepsilon>0,\quad q>0,\quad 1-qT>0 \tag{8.51}$$

参考控制律式(3.45)的推导过程，由式(8.50)、式(8.51)得基于指数趋近律的滑模控制器为

$$u_s(k)=(\boldsymbol{C}\boldsymbol{B})^{-1}[\boldsymbol{C}(\boldsymbol{R}(k+1)-\boldsymbol{R}(k))-\boldsymbol{C}(\boldsymbol{A}-\boldsymbol{I})\boldsymbol{x}(k)-\mathrm{d}s(k)] \tag{8.52}$$

灰色滑模控制包括灰色估计和灰色滑模补偿两个阶段。

2. 灰色估计

灰色系统的研究方法之一，就是将原始数据进行处理，称为数的“生成”，其中累加生成由于能够弱化随机性、增强规律性，因而它在灰色系统建模中具有特殊的地位。按灰色系统理论，采用累加生成方法，可建立灰色预测模型，实现对未知参数和外加干扰的预测。

令 $\boldsymbol{x}^{(0)}$ 为原始的离散数列：

$$\boldsymbol{x}^{(0)} = (\boldsymbol{x}^{(0)}(1)\ \boldsymbol{x}^{(0)}(2)\ \cdots\ \boldsymbol{x}^{(0)}(n)) \tag{8.53}$$

若

$$\boldsymbol{x}^{(1)}(k_1) = \sum_{m=1}^{k} \boldsymbol{x}^{(0)}(m) \tag{8.54}$$

则称 $\boldsymbol{x}^{(1)}(k_1)$为 $\boldsymbol{x}^{(0)}(k)$的累加生成。

利用 $\boldsymbol{x}(k)=\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix}$可得到 $\boldsymbol{x}^{(0)}(k)$，利用式(8.54)可得到 $\boldsymbol{x}_i^{(0)}(k)$的累加生成 $\boldsymbol{x}_i^{(1)}(k_1)$，其中 $i=1,2,\cdots,n$，$k=1,2,\cdots,N$，$k_1=1,2,\cdots,N-2$，n 为系统阶数。

由式(8.47)可得

$$D(x,k) = (\boldsymbol{B})^{-1}(\boldsymbol{x}(k+1) - \boldsymbol{Ax}(k) - \boldsymbol{Bu}(k)) \tag{8.55}$$

其中 $u(k)$为指数趋近律的滑模控制，即 $u(k)=u_s(k)$。

利用式(8.55)，可得到关于 $D(x,k)$ 的离散系列数据 $D^{(0)}(k)$，从而按 $D^{(1)}(k_1) = \sum_{m=1}^{k} D^{(0)}(m)$ 可得到累加生成 $D^{(1)}(k_1)$，$k=1,2,\cdots,N$，$k_1=1,2,\cdots,N-2$。

根据最小二乘法，若($\boldsymbol{BB}^{\mathrm{T}}\boldsymbol{BB}$)可逆，则灰色辨识结果为

$$\hat{\boldsymbol{V}}^{\mathrm{T}} = (\boldsymbol{BB}^{\mathrm{T}}\boldsymbol{BB})^{-1}\boldsymbol{BB}^{\mathrm{T}}D^{(1)} \tag{8.56}$$

其中

$$\hat{\boldsymbol{V}} = (\hat{V}_1\quad \hat{V}_2\quad \cdots\quad \hat{V}_n\quad \hat{d})^{\mathrm{T}} \tag{8.57}$$

$$\boldsymbol{BB} = \begin{bmatrix} x_1^{(1)}(2) & \cdots & x_n^{(1)}(2) & 1 \\ x_1^{(1)}(3) & \cdots & x_n^{(1)}(3) & 2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(1)}(N) & \cdots & x_n^{(1)}(N) & N-2 \end{bmatrix} \tag{8.58}$$

其中$|\det(\boldsymbol{BB}^{\mathrm{T}}\boldsymbol{BB})|>\varepsilon>0$，$x_i^{(1)}(k_1)$为 $x_i^{(0)}(k)$的累加生成。$i=1,2,\cdots,n$，$k=1,2,\cdots,N$，$k_1=1,2,\cdots,N-2$，n 为系统的阶数。

按灰色预测理论，经过 $N=n+3$ 步的灰色辨识，就可实现对干扰参数的辨识。

3. 灰色滑模补偿

利用干扰参数可实现对干扰的有效补偿。灰色补偿控制器为

$$u_c = -\left(\sum_{i=1}^{n} \hat{V}_i x_i + \hat{d}\right) \tag{8.59}$$

此时总的滑模控制器为

$$u = u_s + u_c \tag{8.60}$$

8.4.3 仿真实例

被控对象为二阶传递函数：

$$G(s) = \frac{133}{s^2 + 25s}$$

采样时间取 $T=0.001\mathrm{s}$，被控对象的离散化方程为

$$x(k+1) = Ax(k) + B[u(k) + D(x,k)]$$

其中 $A = \begin{bmatrix} 1 & 0.00098760351887 \\ 0 & 0.97530991202833 \end{bmatrix}$，$B = \begin{bmatrix} 0.00006594927963 \\ 0.13135126800927 \end{bmatrix}$，$D(x,k) = V_1x_1(k) + V_2x_2(k) + d(k)$。

假设系统的不确定部分加到控制输入信号上，取实际干扰参数为

$$V = [1.5 \quad -1.5], d(k) = 0.15$$

设指令为正弦信号：

$$r(k) = 0.50\sin(6\pi kT)$$

在滑模控制器中，取 $C=[15 \quad 1]$，$\varepsilon=0.5$，$q=35$。

仿真包含以下两个部分：

(1) 灰色预测

运行程序 chap8_5.m，对二阶传递函数，$n=2$。使用灰色 GM(0,N)模型，取运行步骤为 $N=n+3=5$。

利用 $k=1,2,\cdots,N$ 原始数据序列得到 $k_1=1,2,\cdots,N-2$ 累加数据序列，从而得到矩阵 $\boldsymbol{BB}$ 为

$$\boldsymbol{BB} = \begin{bmatrix} 0.0002 & 0.4920 & 1.0000 \\ 0.0012 & 1.3556 & 2.0000 \\ 0.0031 & 2.5001 & 3.0000 \end{bmatrix}$$

利用式(8.56)，可得干扰参数估计结果为：

$$\hat{V} = [1.50000000038563 \quad -1.50000000000110 \quad 0.15000000000010]$$

即 $\hat{V}_1=1.50000000038563$，$\hat{V}_2=-1.50000000000110$，$\hat{d}=0.15000000000010$。

(2) 采用灰色预测器补偿的控制

运行程序 chap8_6.m。取 $M=2$，不加灰色补偿，只采用控制律式(8.52)，仿真结果如图 8-18 所示，表明只使用滑模控制器不能克服干扰。取 $M=1$，使用灰色补偿滑模控制律式(8.59)及仿真结果见图 8-19 和图 8-20，表明采用灰色补偿可有效地克服不确定干扰。

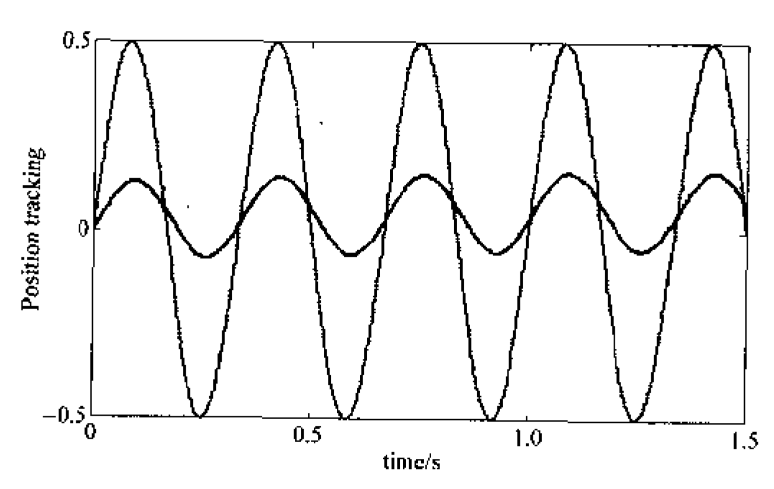


图 8-18　无灰色补偿的位置跟踪

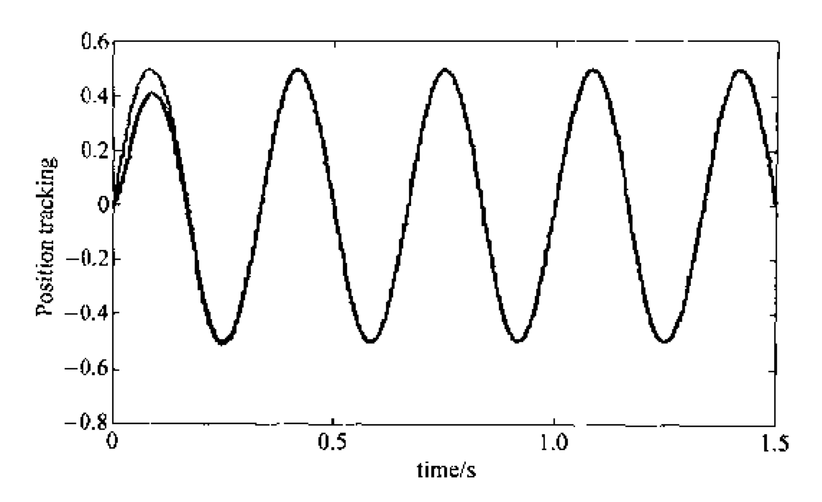


图 8-19 具有灰色补偿的位置跟踪

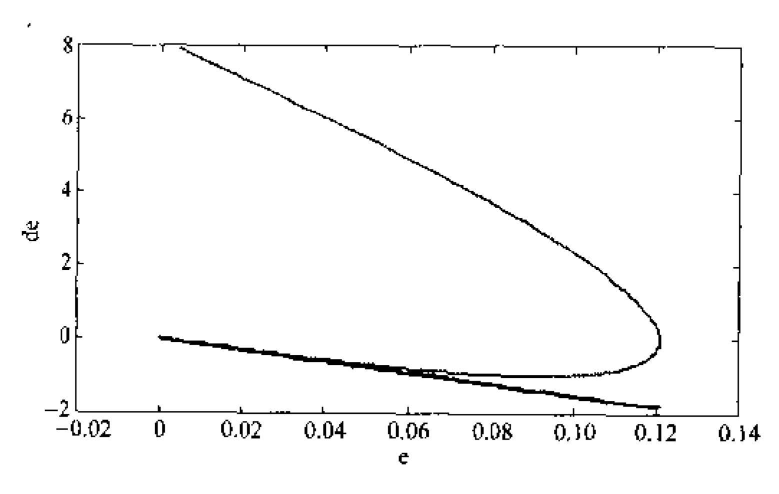


图 8-20 相轨迹

仿真程序：

(1) 灰色预测程序：chap8_5.m

```
% Grey model prediction
clear all;
close all;

ts = 0.001;
n = 2;

I = eye(n);
N = n + 3;
c = 15;
Ce = [c 1];

a = 25;b = 133;
A1 = [0 1;0 - a];
B1 = [0;b];
```

```
C1 = [25 0];
D1 = [0];
[A,B,C,D] = c2dm(A1,B1,C1,D1,ts,'z');

V = [1.5  -1.5];d = 0.15;

x_1 = [0;0];
r_2 = 0;r_1 = 0;

for k = 1:1:N
time(k) = k * ts;

r(k) = 0.5 * (sin(6 * pi * k * ts));
r1(k) = r(k);
r(k + 1) = 0.5 * (sin(6 * pi * (k + 1) * ts));
dr(k) = 0.5 * 6 * pi * cos(6 * pi * k * ts);
dr(k + 1) = 0.5 * 6 * pi * cos(6 * pi * (k + 1) * ts);

x1(k) = x_1(1);
x2(k) = x_1(2);

e(k) = r(k) - x1(k);
de(k) = dr(k) - x2(k);

s(k) = Ce * [e(k);de(k)];

% Exponential velocity trending law
eq = 0.5;
q = 35;
ds(k) = - eq * ts * sign(s(k)) - q * ts * s(k);

xr1 = ([r(k + 1);dr(k + 1)] - [r(k);dr(k)]);      % r(k + 1)
u(k) = inv(Ce * B) * (Ce * xr1 - Ce * (A - I) * [x1(k);x2(k)] - ds(k));

% Plant
  DD = V * x_1 + d;
  x = A * x_1 + B * u(k) + B * DD;
  x_1 = x;

  r_2 = r_1;r_1 = r(k);
end

% Grey prediction
    xx1(1) = x1(2);xx2(1) = x2(2);
    BB = [xx1(1) xx2(1) 1];

for k1 = 2:1:N - 2
    xx1(k1) = xx1(k1 - 1) + x1(k1 + 1);
```

```
    xx2(k1) = xx2(k1 - 1) + x2(k1 + 1);
    BB = [BB;xx1(k1) xx2(k1) k1];
end

for k1 = 1:1:N - 1
   DDD(k1) = 1/B * ([x1(k1 + 1);x2(k1 + 1)] - A * [x1(k1);x2(k1)]) - u(k1);
end
D1(1) = DDD(2);

for k1 = 2:1:N - 2
   D1(k1) = D1(k1 - 1) + DDD(k1 + 1);
end

ab = abs(det(BB' * BB))
V1 = inv(BB' * BB) * BB' * D1';
Vp = V1'
```

(2) 灰色滑模控制程序:chap8_6.m

```
%Discrete Sliding Mode Control with grey model prediction
clear all;
close all;

ts = 0.001;
n = 2;
N1 = 1500;

I = eye(n);
N = n + 3;
c = 15;
Ce = [c 1];

a = 25;b = 133;
A1 = [0 1;0 -a];
B1 = [0;b];
C1 = [25 0];
D1 = [0];
[A,B,C,D] = c2dm(A1,B1,C1,D1,ts,'z');

V = [1.5 -1.5];d = 0.15;
Vp = [1.5000 -1.5000 0.1500];

x_1 = [0;0];
r_2 = 0;r_1 = 0;

for k = 1:1:N1
   time(k) = k * ts;

   r(k) = 0.5 * (sin(6 * pi * k * ts));
```

```
    r1(k) = r(k);
    r(k+1) = 0.5*(sin(6*pi*(k+1)*ts));
    dr(k) = 0.5*6*pi*cos(6*pi*k*ts);
    dr(k+1) = 0.5*6*pi*cos(6*pi*(k+1)*ts);

    x1(k) = x_1(1);
    x2(k) = x_1(2);

 % Control law
uc(k) = -(Vp(1)*x_1(1)+Vp(2)*x_1(2)+Vp(3));    % Grey Compensation
 % uc(k) = 0;    % No Grey Compensation

e(k) = r(k) - x1(k);
de(k) = dr(k) - x2(k);

s(k) = Ce*[e(k);de(k)];

 % Exponential velocity trending law
eq = 0.5;
q = 35;
ds(k) = -eq*ts*sign(s(k)) - q*ts*s(k);

xr1 = ([r(k+1);dr(k+1)]-[r(k);dr(k)]);    % r(k+1)
us(k) = inv(Ce*B)*(Ce*xr1-Ce*(A-I)*[x1(k);x2(k)]-ds(k));

u(k) = us(k) + uc(k);

 % Plant
   DD = V*x_1+d;
   x = A*x_1+B*u(k)+B*DD;
   x_1 = x;
   r_2 = r_1;r_1 = r(k);
end
figure(1);
plot(time,r1,'r',time,x1,'b');
xlabel('time(s)');ylabel('position tracking');
figure(2);
plot(time,u);
xlabel('time(s)');ylabel('control input');
figure(3);
plot(e,de,'r',e,-c*e,'b');
xlabel('e');ylabel('de');
figure(4);
plot(time,s);
xlabel('time(s)');ylabel('s');
```

参 考 文 献

1. Atsuo Kawamura, Hiroshi Itoh, Kiyoshi Sakamoto. Chattering reduction of disturbance observer based sliding mode control. IEEE Transactions on Industry Applications, 1994, 30(2): 456～461
2. Yongsoon Eun, Jung-Ho Kim, Kwangsoo Kim, Dong-Il Cho. Discrete-time variable structure controller with a decoupled disturbance compensator and its application to a CNC servomechanism. IEEE Transactions on Control Systems Technology, 1999, 7(4): 414～423
3. 刘金琨. 先进 PID 控制及其 Matlab 仿真(第 2 版). 北京:电子工业出版社,2004
4. 邓聚龙. 灰色控制系统. 武汉:华中理工大学出版社,1985
5. 姚琼荟, 黄继起, 吴汗松. 变结构控制系统. 重庆:重庆大学出版社,1997

第9章　Terminal 滑模控制

在普通的滑模控制中，通常选择一个线性的滑动平面，使系统到达滑动模态后，跟踪误差渐近地收敛到零。渐近收敛的速度可以通过调整滑模面参数来实现，但无论如何状态跟踪误差都不会在有限时间内收敛到零。

近年来，为了获得更好的性能，一些学者提出了 Terminal 滑模控制策略。所谓 Terminal 滑模控制，就是在滑动超平面的设计中引入了非线性函数，构造 Terminal 滑模面，使得在滑模面上跟踪误差能够在有限时间内收敛到零。

9.1　一种高阶 MIMO 非线性系统的 Terminal 滑模控制

9.1.1　系统描述

考虑如下一类 n 阶多输入多输出非线性系统：

$$\boldsymbol{y}^{(n)} = \boldsymbol{f}(\boldsymbol{y}^{(n-1)},\cdots,\dot{\boldsymbol{y}},\boldsymbol{y},t) + \Delta\boldsymbol{f}(\boldsymbol{y}^{(n-1)},\cdots,\dot{\boldsymbol{y}},\boldsymbol{y},t) + \boldsymbol{b}(\boldsymbol{y}^{(n-1)},\cdots,\dot{\boldsymbol{y}},\boldsymbol{y},t)\boldsymbol{u} + \boldsymbol{d}(t) \tag{9.1}$$

式中 $\boldsymbol{y}\in\mathbf{R}^m$ 为输出向量；$\boldsymbol{u}\in\mathbf{R}^m$ 为控制输入向量；$\boldsymbol{f}\in\mathbf{R}^m$ 和 $\boldsymbol{b}\in\mathbf{R}^{m\times m}$ 为已知的系统状态非线性函数矩阵，$\mathrm{rank}(\boldsymbol{b})=m$；$\Delta\boldsymbol{f}$ 和 $\boldsymbol{d}(t)$ 分别为被控对象的不确定性和外部扰动。

令 $\boldsymbol{x}_1=\boldsymbol{y},\boldsymbol{x}_2=\dot{\boldsymbol{y}},\cdots,\boldsymbol{x}_n=\boldsymbol{y}^{(n-1)}$，则系统(9.1)可写成如下形式：

$$\begin{cases}\dot{\boldsymbol{x}}_1 = \boldsymbol{x}_2\\ \quad\vdots\\ \dot{\boldsymbol{x}}_n = \boldsymbol{f}(\boldsymbol{X},t)+\Delta\boldsymbol{f}(\boldsymbol{X},t)+\boldsymbol{b}(\boldsymbol{X},t)\boldsymbol{u}+\boldsymbol{d}(t)\end{cases} \tag{9.2}$$

其中 $\boldsymbol{X}=[\boldsymbol{x}_1^{\mathrm{T}}\ \boldsymbol{x}_2^{\mathrm{T}}\cdots\boldsymbol{x}_n^{\mathrm{T}}]^{\mathrm{T}}=[\boldsymbol{x}_1^{\mathrm{T}}\ \dot{\boldsymbol{x}}_1^{\mathrm{T}}\cdots\boldsymbol{x}_1^{(n-1)\mathrm{T}}]^{\mathrm{T}}$。

不确定性 $\Delta\boldsymbol{f}(\boldsymbol{X},t)$ 和外部扰动 $\boldsymbol{d}(t)$ 满足下面的条件：

$$|\Delta\boldsymbol{f}(\boldsymbol{X},t)|\leqslant F(\boldsymbol{X},t),\ |\boldsymbol{d}(t)|\leqslant D(t) \tag{9.3}$$

其中 $F(\boldsymbol{X},t)$ 和 $D(t)$ 是两个非负的函数。

9.1.2　Terminal 滑模控制器设计

1. 切换面的设计

通过设计控制律，使系统的状态 $\boldsymbol{X}=[\boldsymbol{x}_1^{\mathrm{T}}\ \boldsymbol{x}_2^{\mathrm{T}}\cdots\boldsymbol{x}_n^{\mathrm{T}}]^{\mathrm{T}}=[\boldsymbol{x}_1^{\mathrm{T}}\ \dot{\boldsymbol{x}}_1^{\mathrm{T}}\cdots\boldsymbol{x}_1^{(n-1)\mathrm{T}}]^{\mathrm{T}}$ 在有限时间内实现对期望状态 $\boldsymbol{X}_{\mathrm{d}}=[\boldsymbol{x}_{1\mathrm{d}}^{\mathrm{T}}\ \boldsymbol{x}_{2\mathrm{d}}^{\mathrm{T}}\cdots\boldsymbol{x}_{n\mathrm{d}}^{\mathrm{T}}]^{\mathrm{T}}=[\boldsymbol{x}_{1\mathrm{d}}^{\mathrm{T}}\ \dot{\boldsymbol{x}}_{1\mathrm{d}}^{\mathrm{T}}\cdots\boldsymbol{x}_{1\mathrm{d}}^{(n-1)\mathrm{T}}]^{\mathrm{T}}$ 的跟踪。

定义误差向量为

$$\boldsymbol{E}=\boldsymbol{X}-\boldsymbol{X}_{\mathrm{d}}=[\boldsymbol{e}^{\mathrm{T}}\ \dot{\boldsymbol{e}}^{\mathrm{T}}\cdots\boldsymbol{e}^{(n-1)\mathrm{T}}]^{\mathrm{T}} \tag{9.4}$$

其中 $\boldsymbol{e}=\boldsymbol{x}_1-\boldsymbol{x}_{1d}=[e_1\ e_2\cdots e_m]^{\mathrm{T}}$。

滑模面方程设计为

$$\boldsymbol{\sigma}(\boldsymbol{X},t)=\boldsymbol{CE}-\boldsymbol{W}(t) \tag{9.5}$$

式中 $\boldsymbol{C}=[\boldsymbol{C}_1\boldsymbol{C}_2\cdots\boldsymbol{C}_n]$为矩阵，$\boldsymbol{C}_i=(c_{i1},c_{i2},\cdots,c_{im})$，$c_{ij}(i=1,2,\cdots,n;j=1,2,\cdots,m)$是正常数；$\boldsymbol{W}(t)=\boldsymbol{CP}(t)$，$\boldsymbol{P}(t)=[\boldsymbol{p}(t)^{\mathrm{T}}\dot{\boldsymbol{p}}(t)^{\mathrm{T}}\cdots\boldsymbol{p}^{(n-1)}(t)^{\mathrm{T}}]^{\mathrm{T}}$。

令 $\boldsymbol{p}(t)=[p_1(t)p_2(t)\cdots p_m(t)]^{\mathrm{T}}$，且 $p_i(t)$满足下面的假设。

假设 1 $p_i(t):\mathbf{R}_+\to\mathbf{R}$，$p_i(t)\in\boldsymbol{C}^n[0,\infty)$，$\dot{p}_i,\cdots,p_i^{(n)}\in L^{\infty}$。对于某个常数 $T>0$，$p_i(t)$是在时间段$[0,T]$上有界的，并且 $p_i(0)=e_i(0)$，$\dot{p}_i(0)=\dot{e}_i(0)$，$\cdots$，$p_i^{(n)}(0)=e_i^{(n)}(0)$。$\boldsymbol{C}^n[0,\infty)$表示定义在$[0,\infty)$的所有 n 阶可微的连续函数，$i=1,2,\cdots,m$。

选取函数 $p_i(t)$为

$$p_i(t)=\begin{cases}\sum\limits_{k=0}^{n}\dfrac{1}{k!}e_i(0)^{(k)}t^k+\sum\limits_{j=0}^{n}\left(\sum\limits_{l=0}^{n}\dfrac{a_{jl}}{T^{j-l+n+1}}e_i(0)^{(l)}\right)\cdot t^{j+n+1} & 0\leqslant t\leqslant T\\ 0 & t>T\end{cases} \tag{9.6}$$

其中参数 a_{jl}可通过假设 1 中的条件求得。

以三阶系统为例，给出函数 $p_i(t)$的设计过程。$n=3$ 时，函数 $p_i(t)$为如下形式：

$$p_i(t)=\begin{cases}e_i(0)+\dot{e}_i(0)t+\dfrac{1}{2}\ddot{e}_i(0)t^2+\dfrac{1}{6}\dddot{e}_i(0)t^3+\\ \left[\dfrac{a_{00}}{T^4}e_i(0)+\dfrac{a_{01}}{T^3}\dot{e}_i(0)+\dfrac{a_{02}}{T^2}\ddot{e}_i(0)+\dfrac{a_{03}}{T}\dddot{e}_i(0)\right]t^4+\\ \left[\dfrac{a_{10}}{T^5}e_i(0)+\dfrac{a_{11}}{T^4}\dot{e}_i(0)+\dfrac{a_{12}}{T^3}\ddot{e}_i(0)+\dfrac{a_{13}}{T^2}\dddot{e}_i(0)\right]t^5+\\ \left[\dfrac{a_{20}}{T^6}e_i(0)+\dfrac{a_{21}}{T^5}\dot{e}_i(0)+\dfrac{a_{22}}{T^4}\ddot{e}_i(0)+\dfrac{a_{23}}{T^3}\dddot{e}_i(0)\right]t^6+\\ \left[\dfrac{a_{30}}{T^7}e_i(0)+\dfrac{a_{31}}{T^6}\dot{e}_i(0)+\dfrac{a_{32}}{T^5}\ddot{e}_i(0)+\dfrac{a_{33}}{T^4}\dddot{e}_i(0)\right]t^7\\ \qquad\qquad \text{if}\quad 0\leqslant t\leqslant T\\ 0 \qquad\qquad \text{if}\quad t>T\end{cases} \tag{9.7}$$

由式(9.7)可得 $\dot{p}_i(t)$，$\ddot{p}_i(t)$和 $\dddot{p}_i(t)$的表达式。根据假设 1，函数 $p_i(t)$是一个在 $t=T$ 时刻三阶可微的连续函数，当 $t=T$ 时，函数 $p_i(t)$，$\dot{p}_i(t)$，$\ddot{p}_i(t)$和 $\dddot{p}_i(t)$的值均为零。

令参数 a_{jl}满足下面的 4 组的四元一次方程组，就可以使 $p_i(T)$，$\dot{p}_i(T)$，$\ddot{p}_i(T)$和 $\dddot{p}_i(T)$等于零，即

$$\begin{cases}1+a_{00}+a_{10}+a_{20}+a_{30}=0\\ 4a_{00}+5a_{10}+6a_{20}+7a_{30}=0\\ 12a_{00}+20a_{10}+30a_{20}+42a_{30}=0\\ 24a_{00}+60a_{10}+120a_{20}+210a_{30}=0\end{cases}$$

$$\begin{cases}1+a_{01}+a_{11}+a_{21}+a_{31}=0\\ 1+4a_{01}+5a_{11}+6a_{21}+7a_{31}=0\\ 12a_{01}+20a_{11}+30a_{21}+42a_{31}=0\\ 24a_{01}+60a_{11}+120a_{21}+210a_{31}=0\end{cases}$$

$$\begin{cases}0.5 + a_{02} + a_{12} + a_{22} + a_{32} = 0 \\ 1 + 4a_{02} + 5a_{12} + 6a_{22} + 7a_{32} = 0 \\ 1 + 12a_{02} + 20a_{12} + 30a_{22} + 42a_{32} = 0 \\ 24a_{02} + 60a_{12} + 120a_{22} + 210a_{32} = 0\end{cases}$$

$$\begin{cases}\dfrac{1}{6} + a_{03} + a_{13} + a_{23} + a_{33} = 0 \\ 0.5 + 4a_{03} + 5a_{13} + 6a_{23} + 7a_{33} = 0 \\ 1 + 12a_{03} + 20a_{13} + 30a_{23} + 42a_{33} = 0 \\ 1 + 24a_{03} + 60a_{13} + 120a_{23} + 210a_{33} = 0\end{cases}$$

解上述方程组，可得参数 a_{jl}（$j=0,1,2,3$；$l=0,1,2,3$）的值：

$$\begin{cases}a_{00} = -35 \\ a_{10} = 84 \\ a_{20} = -70 \\ a_{30} = 20\end{cases} \quad \begin{cases}a_{01} = -20 \\ a_{11} = 45 \\ a_{21} = -36 \\ a_{31} = 10\end{cases} \quad \begin{cases}a_{02} = -5 \\ a_{12} = 10 \\ a_{22} = -7.5 \\ a_{32} = 2\end{cases} \quad \begin{cases}a_{03} = -\dfrac{2}{3} \\ a_{13} = 1 \\ a_{23} = -\dfrac{2}{3} \\ a_{33} = \dfrac{1}{6}\end{cases} \tag{9.8}$$

同理，可得到 n 阶系统相应的参数 a_{jl}，从而确定 n 阶系统的 Terminal 滑模面。

2. Terminal 滑模控制器的设计

由式（9.5）得

$$\begin{aligned}\dot{\boldsymbol{\sigma}}(\boldsymbol{X},t) &= \boldsymbol{C}\dot{\boldsymbol{E}} - \boldsymbol{C}\dot{\boldsymbol{P}}(t) \\ &= \boldsymbol{C}\cdot[\dot{e}^{\mathrm{T}}\ddot{e}^{\mathrm{T}}\cdots e^{(n)\mathrm{T}}]^{\mathrm{T}} - \boldsymbol{C}\cdot[\dot{\boldsymbol{p}}(t)^{\mathrm{T}}\ddot{\boldsymbol{p}}(t)^{\mathrm{T}}\cdots\boldsymbol{p}^{(n)}(t)^{\mathrm{T}}]^{\mathrm{T}} \\ &= \boldsymbol{C}_n[\boldsymbol{e}^{(n)} - \boldsymbol{p}^{(n)}(t)] + \sum_{k=1}^{n-1}\boldsymbol{C}_k[\boldsymbol{e}^{(k)} - \boldsymbol{p}^{(k)}(t)] \\ &= \boldsymbol{C}_n[\boldsymbol{x}_1^{(n)} - \boldsymbol{x}_{1d}^{(n)} - \boldsymbol{p}^{(n)}(t)] + \sum_{k=1}^{n-1}\boldsymbol{C}_k[\boldsymbol{e}^{(k)} - \boldsymbol{p}^{(k)}(t)] \\ &= \boldsymbol{C}_n[\dot{\boldsymbol{x}}_n - \boldsymbol{x}_{1d}^{(n)} - \boldsymbol{p}^{(n)}(t)] + \sum_{k=1}^{n-1}\boldsymbol{C}_k[\boldsymbol{e}^{(k)} - \boldsymbol{p}^{(k)}(t)]\end{aligned}$$

即

$$\begin{aligned}\dot{\boldsymbol{\sigma}}(\boldsymbol{X},t) = &\boldsymbol{C}_n[\boldsymbol{f}(\boldsymbol{X},t) + \Delta\boldsymbol{f}(\boldsymbol{X},t) + \boldsymbol{b}(\boldsymbol{X},t)\boldsymbol{u} + \boldsymbol{d}(t) \\ &- \boldsymbol{x}_{1d}^{(n)} - \boldsymbol{p}(t)^{(n)}] + \sum_{k=1}^{n-1}\boldsymbol{C}_k[\boldsymbol{e}^{(k)} - \boldsymbol{p}(t)^{(k)}]\end{aligned}$$

设计 Lyapunov 函数为

$$V = \frac{1}{2}\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{\sigma}$$

则

$$\begin{aligned}\dot{V} = \boldsymbol{\sigma}^{\mathrm{T}}\dot{\boldsymbol{\sigma}} = &\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{C}_n\{\boldsymbol{f}(\boldsymbol{X},t) - \boldsymbol{x}_{1d}^{(n)} - \boldsymbol{p}(t)^{(n)} + \boldsymbol{C}_n^{-1}\sum_{k=1}^{n-1}\boldsymbol{C}_k[\boldsymbol{e}^{(k)} - \boldsymbol{p}(t)^{(k)}]\} + \\ &\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{C}_n\boldsymbol{b}(\boldsymbol{X},t)\boldsymbol{u} + \boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{C}_n[\Delta\boldsymbol{f}(\boldsymbol{X},t) + \boldsymbol{d}(t)] \leqslant \boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{C}_n \times\end{aligned}$$

$$\{f(\boldsymbol{X},t)-x_{1d}^{(n)}-p(t)^{(n)}+\boldsymbol{C}_n^{-1}\sum_{k=1}^{n-1}C_k[e^{(k)}-p(t)^{(k)}]\}+$$

$$\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{C}_n\boldsymbol{b}(\boldsymbol{X},t)\boldsymbol{u}+\|\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{C}_n\|\ \|\Delta\boldsymbol{f}(\boldsymbol{X},t)+\boldsymbol{d}(t)\|$$

控制器设计为

$$\boldsymbol{u}(t)=-\boldsymbol{b}(\boldsymbol{X},t)^{-1}\{\boldsymbol{f}(\boldsymbol{X},t)-\boldsymbol{x}_{1d}^{(n)}-\boldsymbol{p}(t)^{(n)}+\boldsymbol{C}_n^{-1}\sum_{k=1}^{n-1}\boldsymbol{C}_k(\boldsymbol{e}^{(k)}-\boldsymbol{p}(t)^{(k)})\}$$

$$-\boldsymbol{b}(\boldsymbol{X},t)^{-1}\frac{\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}}{\|\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}\|}\{F(\boldsymbol{X},t)+D(t)+K\} \tag{9.9}$$

其中 K 为正常数。

将式(9.9)代入 $\dot{V}$ 得

$$\dot{V}\leqslant\|\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{C}_n\|\ \|\Delta\boldsymbol{f}(\boldsymbol{X},t)+\boldsymbol{d}(t)\|-\frac{\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{C}_n\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}}{\|\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}\|}\{F(\boldsymbol{X},t)+D(t)+K\}$$

由于

$$\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{C}_n\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}=\|\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}\|^2$$

则

$$\begin{aligned}\dot{V}&\leqslant\|\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}\|\{\|\Delta\boldsymbol{f}(\boldsymbol{X},t)+\boldsymbol{d}(t)\|-[F(\boldsymbol{X},t)+D(t)]\}-K\|\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}\|\\&=\|\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}\|\{[\|\Delta\boldsymbol{f}(\boldsymbol{X},t)\|-F(\boldsymbol{X},t)]+[\|\boldsymbol{d}(t)\|-D(t)]\}-K\|\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}\|\\&\leqslant-K\|\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}\|<0\qquad(|\sigma|\neq0)\end{aligned} \tag{9.10}$$

根据假设 1 和 Terminal 滑模面方程(9.5)得

$$\begin{aligned}\boldsymbol{\sigma}(\boldsymbol{X},0)&=\boldsymbol{C}\boldsymbol{E}(0)-\boldsymbol{W}(0)=\boldsymbol{C}[\boldsymbol{E}(0)-\boldsymbol{P}(0)]\\&=\boldsymbol{C}\{[\boldsymbol{e}(0)^{\mathrm{T}}\dot{\boldsymbol{e}}(0)^{\mathrm{T}}\cdots\boldsymbol{e}(0)^{(n-1)\mathrm{T}}]^{\mathrm{T}}-[\boldsymbol{p}(0)^{\mathrm{T}}\dot{\boldsymbol{p}}(0)^{\mathrm{T}}\cdots\boldsymbol{p}^{(n-1)}(0)^{\mathrm{T}}]^{\mathrm{T}}\}\\&=0\end{aligned} \tag{9.11}$$

即系统的初始状态已经在滑模面上，消除了滑模控制的到达阶段，确保了闭环系统的全局鲁棒性和稳定性。

由于系统具有全局鲁棒性，即 $\boldsymbol{\sigma}(\boldsymbol{X},t)=0$ 或 $\boldsymbol{E}(t)=\boldsymbol{P}(t)$。故通过选择 Terminal 滑模面中的函数 $\boldsymbol{P}(T)=0$，实现 $\boldsymbol{E}(T)=0$，从而保证跟踪误差在有限时间 T 内收敛至零。

为了降低抖振，采用连续函数向量 $\boldsymbol{S}_\delta$ 代替 $\dfrac{\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}}{\|\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}\|}$：

$$\boldsymbol{S}_\delta=\frac{\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}}{\|\boldsymbol{C}_n^{\mathrm{T}}\boldsymbol{\sigma}\|+\delta} \tag{9.12}$$

$$\delta=\delta_0+\delta_1\|\boldsymbol{e}\| \tag{9.13}$$

其中 δ_0，δ_1 为两个正常数。

9.1.3 仿真实例之一：SISO 系统的 Terminal 滑模控制

考虑二阶 SISO 系统：

$$\begin{cases}\dot{x}_1=x_2\\\dot{x}_2=-25x_1+133u+3.0\sin(2\pi t)\end{cases}$$

其中 $f(x,t)=-25x_1$，$b(x,t)=133$，$d(t)=3.0\sin(2\pi t)$。

对于二阶 SISO 系统，$m=1,n=2$ 时，$p_i(t)$的表达式为

$$p_1(t)=\begin{cases}\sum_{k=0}^{2}\frac{1}{k!}e_i(0)^{(k)}t^k+\sum_{j=0}^{2}\left(\sum_{l=0}^{2}\frac{a_{jl}}{T^{j-l+3}}e_i(0)^{(l)}\right)\cdot t^{j+3} & 0\leqslant t\leqslant T\\ 0 & t>T\end{cases}$$

当 $i=1,j=0,1,2$ 时，函数 $p_1(t)$及其导数可写为

$$p_1(t)=e_1(0)+\dot{e}_1(0)t+\frac{1}{2}\ddot{e}_1(0)t^2+\left(\frac{a_{00}}{T^3}e_1(0)+\frac{a_{01}}{T^2}\dot{e}_1(0)+\frac{a_{02}}{T}\ddot{e}_1(0)\right)t^3+$$
$$\left(\frac{a_{10}}{T^4}e_1(0)+\frac{a_{11}}{T^3}\dot{e}_1(0)+\frac{a_{12}}{T^2}\ddot{e}_1(0)\right)t^4+\left(\frac{a_{20}}{T^5}e_1(0)+\frac{a_{21}}{T^4}\dot{e}_1(0)+\frac{a_{22}}{T^3}\ddot{e}_1(0)\right)t^5$$

$$\dot{p}_1(t)=\dot{e}_1(0)+\ddot{e}_1(0)t+3\left(\frac{a_{00}}{T^3}e_1(0)+\frac{a_{01}}{T^2}\dot{e}_1(0)+\frac{a_{02}}{T}\ddot{e}_1(0)\right)t^2+$$
$$4\left(\frac{a_{10}}{T^4}e_1(0)+\frac{a_{11}}{T^3}\dot{e}_1(0)+\frac{a_{12}}{T^2}\ddot{e}_1(0)\right)t^3+5\left(\frac{a_{20}}{T^5}e_1(0)+\frac{a_{21}}{T^4}\dot{e}_1(0)+\frac{a_{22}}{T^3}\ddot{e}_1(0)\right)t^4$$

$$\ddot{p}_1(t)=\ddot{e}_1(0)+6\left(\frac{a_{00}}{T^3}e_1(0)+\frac{a_{01}}{T^2}\dot{e}_1(0)+\frac{a_{02}}{T}\ddot{e}_1(0)\right)t+$$
$$12\left(\frac{a_{10}}{T^4}e_1(0)+\frac{a_{11}}{T^3}\dot{e}_1(0)+\frac{a_{12}}{T^2}\ddot{e}_1(0)\right)t^2+20\left(\frac{a_{20}}{T^5}e_1(0)+\frac{a_{21}}{T^4}\dot{e}_1(0)+\frac{a_{22}}{T^3}\ddot{e}_1(0)\right)t^3$$

当 $t=T$ 时，$p_1(t)=0$，可得

$$p_1(T)=e_1(0)+\dot{e}_1(0)T+\frac{1}{2}\ddot{e}_1(0)T^2+\left(\frac{a_{00}}{T^3}e_1(0)+\frac{a_{01}}{T^2}\dot{e}_1(0)+\frac{a_{02}}{T}\ddot{e}_1(0)\right)T^3+$$
$$\left(\frac{a_{10}}{T^4}e_1(0)+\frac{a_{11}}{T^3}\dot{e}_1(0)+\frac{a_{12}}{T^2}\ddot{e}_1(0)\right)T^4+\left(\frac{a_{20}}{T^5}e_1(0)+\frac{a_{21}}{T^4}\dot{e}_1(0)+\frac{a_{22}}{T^3}\ddot{e}_1(0)\right)T^5$$
$$=(1+a_{00}+a_{10}+a_{20})e_1(0)+T(1+a_{01}+a_{11}+a_{21})\dot{e}(0)+$$
$$T^2\left(\frac{1}{2}+a_{02}+a_{12}+a_{22}\right)\ddot{e}_1(0)=0$$

则 $p_1(T)=0$ 成立的必要条件为

$$\begin{cases}1+a_{00}+a_{10}+a_{20}=0\\1+a_{01}+a_{11}+a_{21}=0\\0.5+a_{02}+a_{12}+a_{22}=0\end{cases}$$

同理，由 $t=T$ 时，$\dot{p}_1(t)=0$，得 $\ddot{p}_1(t)=0$ 的必要条件为

$$\begin{cases}3a_{00}+4a_{10}+5a_{20}=0\\1+3a_{01}+4a_{11}+5a_{21}=0\\1+3a_{02}+4a_{12}+5a_{22}=0\end{cases}$$

$$\begin{cases}6a_{00}+12a_{10}+20a_{20}=0\\6a_{01}+12a_{11}+20a_{21}=0\\1+6a_{02}+12a_{12}+20a_{22}=0\end{cases}$$

由上述方程组可整理出三个三元一次方程组：

$$\begin{cases} a_{00}+a_{10}+a_{20}=-1 \\ 3a_{00}+4a_{10}+5a_{20}=0 \\ 6a_{00}+12a_{10}+20a_{20}=0 \end{cases}$$

$$\begin{cases} a_{01}+a_{11}+a_{21}=-1 \\ 3a_{01}+4a_{11}+5a_{21}=-1 \\ 6a_{01}+12a_{11}+20a_{21}=0 \end{cases}$$

$$\begin{cases} a_{02}+a_{12}+a_{22}=-0.5 \\ 3a_{02}+4a_{12}+5a_{22}=-1 \\ 6a_{02}+12a_{12}+20a_{22}=-1 \end{cases}$$

上述三个方程组分别可写为以下三种形式：

$$\boldsymbol{A}_1\boldsymbol{x}_1=\boldsymbol{B}_1,\boldsymbol{A}_1=\begin{bmatrix}1&1&1\\3&4&5\\6&12&20\end{bmatrix},\boldsymbol{B}_1=\begin{bmatrix}-1\\0\\0\end{bmatrix}$$

$$\boldsymbol{A}_2\boldsymbol{x}_2=\boldsymbol{B}_2,\boldsymbol{A}_2=\begin{bmatrix}1&1&1\\3&4&5\\6&12&20\end{bmatrix},\boldsymbol{B}_2=\begin{bmatrix}-1\\-1\\0\end{bmatrix}$$

$$\boldsymbol{A}_3\boldsymbol{x}_3=\boldsymbol{B}_3,\boldsymbol{A}_3=\begin{bmatrix}1&1&1\\3&4&5\\6&12&20\end{bmatrix},\boldsymbol{B}_3=\begin{bmatrix}-0.5\\-1\\-1\end{bmatrix}$$

运行初始化子程序 chap9_1int. m，得到上述方程组的解：

$$\begin{cases} a_{00}=-10 \\ a_{10}=15 \\ a_{20}=-6 \end{cases} \quad \begin{cases} a_{01}=-6 \\ a_{11}=8 \\ a_{21}=-3 \end{cases} \quad \begin{cases} a_{02}=-1.5 \\ a_{12}=1.5 \\ a_{22}=-0.5 \end{cases}$$

则得 $p_1(t)$ 的表达式为

$$p_1(t)=\begin{cases} e_0+\dot{e}_0t+\frac{1}{2}\ddot{e}_0t^2-\left(\frac{10}{T^3}e_0+\frac{6}{T^2}\dot{e}_0+\frac{3}{2T}\ddot{e}_0\right)t^3+ \\ \left(\frac{15}{T^4}e_0+\frac{8}{T^3}\dot{e}_0+\frac{3}{2T^2}\ddot{e}_0\right)t^4-\left(\frac{6}{T^5}e_0+\frac{3}{T^4}\dot{e}_0+\frac{1}{2T^3}\ddot{e}_0\right)t^5 & \text{if} \quad 0\leqslant t\leqslant T \\ 0 & \text{if} \quad t>T \end{cases}$$

设位置指令为 $x_d=0.5\sin(2\pi t)$。令 $\mathbf{C}=[4\ 1]$，则 $\sigma(\boldsymbol{X},t)=4e+\dot{e}-4p(t)-\dot{p}(t)$。根据式(9.9)，控制器为

$$\boldsymbol{u}(t)=-\frac{1}{133}\{-25x_2-\ddot{x}_{1d}-\ddot{p}(t)+\boldsymbol{C}_2^{-1}\cdot\boldsymbol{C}_1[\dot{e}-\dot{p}(t)]\}-\frac{1}{133}\frac{C_2\sigma(\boldsymbol{X},t)}{\|C_2\sigma(\boldsymbol{X},t)\|}[D(t)+K]$$

在程序中，$M=1$ 为采用符号函数，$M=2$ 为采用连续函数。在控制器中，取 $\delta_0=0.03$，$\delta_1=5$，$\delta=\delta_0+\delta_1|e|$，$K=10$，$D(t)=3.0$。

系统的初始条件为[0.5 0]。分别取 Terminal 时间为 $T=0.5$ 和 $T=1.0$。取 $M=2$，仿真结果如图 9-1～图 9-3 所示。可见，在 T 时跟踪误差为零，并且通过采用连续函数，降低了抖振。

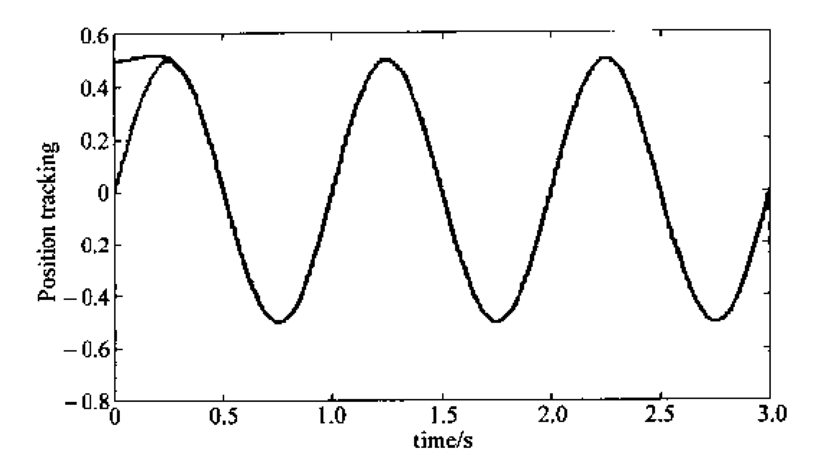


图 9-1　位置跟踪($T=0.5$)

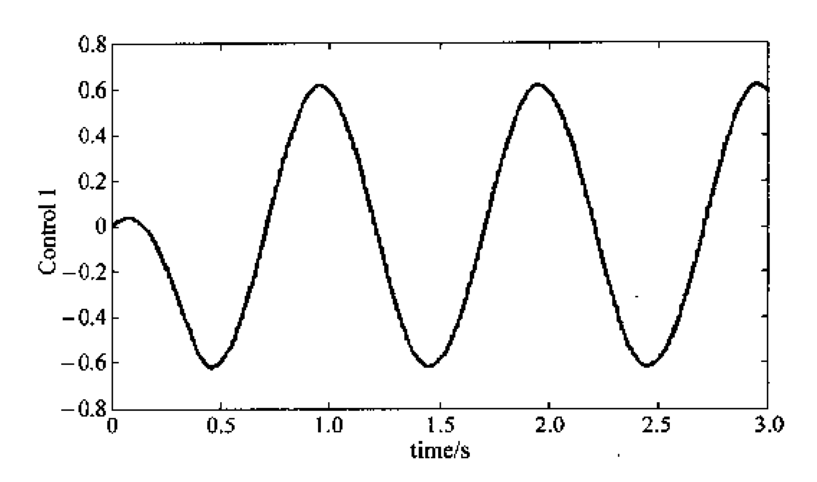


图 9-2　控制输入($T=0.5$)

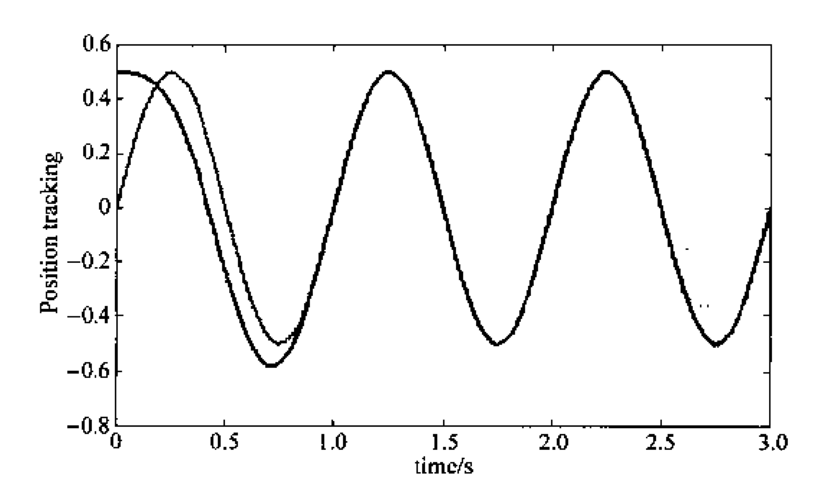


图 9-3　位置跟踪($T=1.0$)

仿真程序：

(1) 主程序(如图 9-4 所示)：chap9_1sim.mdl

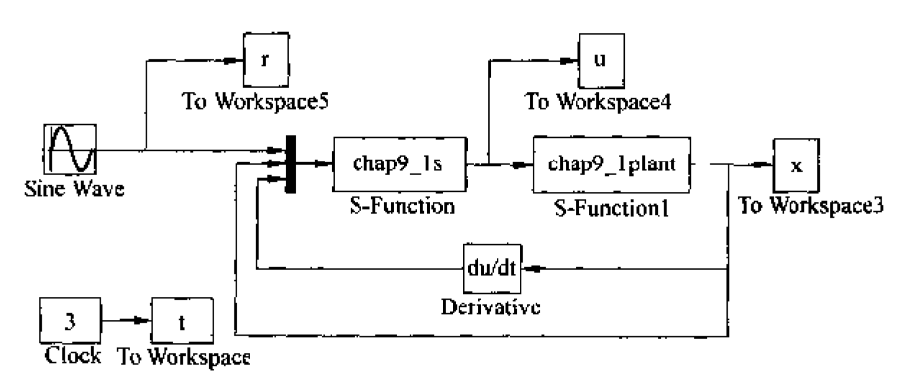


图 9-4 主程序图

(2) 初始化子程序：chap9_1int.m

```
close all;
clear all;

A1 = [1 1 1;3 4 5;6 12 20];
b1 = [ - 1;0;0];
x1 = A1\b1;
A2 = [1 1 1;3 4 5;6 12 20];
b2 = [ - 1; - 1;0];
x2 = A2\b2;
A3 = [1 1 1;3 4 5;6 12 20];
b3 = [ - 1/2; - 1; - 1];
x3 = A3\b3;
```

(3) S 函数控制子程序：chap9_1s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 5;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
```

```
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0 0];

function sys = mdlOutputs(t,x,u)
persistent e0 de0 dde0
T = 1.0;

xd = u(1);
dxd = 0.5 * 2 * pi * cos(2 * pi * t);
ddxd = - 0.5 * (2 * pi)^2 * sin(2 * pi * t);

x = u(2:1:3);
dx = u(4:1:5);
if t == 0
   e0 = x(1);
   de0 = x(2) - pi;
   dde0 = dx(2);
end

C = [5,1];

e = x(1) - xd;
de = x(2) - dxd;
dde = dx(2) - ddxd;

if t< = T
   A0 = - 10/T^3 * e0 - 6/T^2 * de0 - 1.5/T * dde0;
   A1 = 15/T^4 * e0 + 8/T^3 * de0 + 1.5/T^2 * dde0;
   A2 = - 6/T^5 * e0 - 3/T^4 * de0 - 0.5/T^3 * dde0;
   p = e0 + de0 * t + 1/2 * dde0 * t^2 + A0 * t^3 + A1 * t^4 + A2 * t^5;
   dp = de0 + dde0 * t + A0 * 3 * t^2 + A1 * 4 * t^3 + A2 * 5 * t^4;
   ddp = dde0 + A0 * 3 * 2 * t + A1 * 4 * 3 * t^2 + A2 * 5 * 4 * t^3;
else
   p = 0;dp = 0;ddp = 0;
end

rou = (C(1) * e + C(2) * de) - (C(1) * p + C(2) * dp);

delta0 = 0.03;
delta1 = 1.5;
delta = delta0 + delta1 * norm(e);
mrou = norm(rou) + delta;

K = 10;
D = 3.0;
M = 2;
if M == 1
   ut = - 1/133 * ( - 25 * x(2) - ddxd - ddp + inv(C(2)) * C(1) * (de - dp)) - 1/133 * sign(C(2) *
rou) * (D + K);
elseif M == 2
ut = - 1/133 * ( - 25 * x(2) - ddxd - ddp + inv(C(2)) * C(1) * (de - dp)) - 1/133 * rou/mrou * (D +
K);
```

```
end
sys(1) = ut;
```

(4) S 函数被控对象子程序:chap9_1plant. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 2;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0   = [0.5,0];
str = [];
ts   = [0 0];

function sys = mdlDerivatives(t,x,u)
dt = 3.0 * sin(2 * pi * t);

sys(1) = x(2);
sys(2) = -25 * x(2) + 133 * u + dt;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(5) 作图子程序:chap9_1plot. m

```
close all;

figure(1);
plot(t,r(:,1),'r',t,x(:,1),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
```

```
xlabel('time(s)');ylabel('control 1');
```

9.1.4 仿真实例之二：MIMO 系统的 Terminal 滑模控制

考虑如下三输入三输出 MIMO 非线性系统：

$$\dot{x}_{11} = x_{12}$$

$$\dot{x}_{12} = x_{13}$$

$$\dot{x}_{13} = 1.5x_{13}^2\cos(3x_{21}) + (\sin t)\sin(x_{22}^2)\cos(3x_{11}) + u_1$$

$$\dot{x}_{21} = x_{22}$$

$$\dot{x}_{22} = x_{23}$$

$$\dot{x}_{23} = 3x_{22}\sin(x_{31}) + e^{-t}\cos(x_{13})\sin(x_{31}) + u_2$$

$$\dot{x}_{31} = x_{32}$$

$$\dot{x}_{32} = x_{33}$$

$$\dot{x}_{33} = x_{33}\cos(x_{11})\sin(x_{21}) + \cos(t)\sin(x_{23}) + u_3$$

$$\boldsymbol{y}=[x_{11}\,x_{21}\,x_{31}]^T$$

其中$\boldsymbol{f}(\boldsymbol{X},t)=\begin{bmatrix}-1.5x_{13}^2\cos(3x_{21}) \\ 3x_{22}\sin(x_{31}) \\ x_{33}\cos(x_{11})\sin(x_{21})\end{bmatrix}$， $\boldsymbol{b}(\boldsymbol{X},t)=\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$，

$$\Delta\boldsymbol{f}(\boldsymbol{X},t)=\begin{bmatrix}\sin(t)\sin(x_{22}^2)\cos(3x_{11})\\ e^{-t}\cos(x_{13})\sin(x_{31})\\ \cos(t)\sin(x_{23})\end{bmatrix},\boldsymbol{d}(t)=0。$$

对本系统，$n=3$，$m=3$，取

$$\boldsymbol{C}=[\boldsymbol{C}_1\boldsymbol{C}_2\cdots\boldsymbol{C}_n]=[\boldsymbol{C}_1\boldsymbol{C}_2\boldsymbol{C}_3]$$

其中 $\boldsymbol{C}_i=[c_{i1}\,c_{i2}\cdots c_{im}]$，式中 $c_{ij}(i=1,2,\cdots,n;j=1,2,\cdots,m)$为正常数。

取

$$\boldsymbol{C}_1=\mathrm{diag}(c_{11},c_{12},c_{13})=\begin{bmatrix}4&0&0\\0&4&0\\0&0&4\end{bmatrix},\boldsymbol{C}_2=\mathrm{diag}(c_{21},c_{22},c_{23})=\begin{bmatrix}4&0&0\\0&4&0\\0&0&4\end{bmatrix},$$

$$\boldsymbol{C}_3=\mathrm{diag}(c_{31},c_{32},c_{33})=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}。$$

则

$$\boldsymbol{C}=[\boldsymbol{C}_1\boldsymbol{C}_2\boldsymbol{C}_3]=\begin{bmatrix}4&0&0&4&0&0&1&0&0\\0&4&0&0&4&0&0&1&0\\0&0&4&0&0&4&0&0&1\end{bmatrix}$$

由式(9.5)得 Terminal 滑模面为

$$\boldsymbol{\sigma}(\boldsymbol{X},t)=\begin{bmatrix}\sigma_1\\ \sigma_2\\ \sigma_3\end{bmatrix}=\begin{bmatrix}4e_1+4\dot{e}_1+\ddot{e}_1-4p_1-4\dot{p}_1-\ddot{p}_1\\ 4e_2+4\dot{e}_2+\ddot{e}_2-4p_2-4\dot{p}_2-\ddot{p}_2\\ 4e_3+4\dot{e}_3+\ddot{e}_3-4p_3-4\dot{p}_3-\ddot{p}_3\end{bmatrix}$$

针对三阶系统，$n=3$，由式(9.6)得 $p_i(t)(i=1,2,3)$的表达式为

$$p_i(t)=\begin{cases}e_i(0)+\dot{e}_i(0)t+\frac{1}{2}\ddot{e}_i(0)t^2+\frac{1}{6}\dddot{e}_i(0)t^3-\\ \left[\frac{35}{T^4}e_i(0)+\frac{20}{T^3}\dot{e}_i(0)+\frac{5}{T^2}\ddot{e}_i(0)+\frac{2}{3T}\dddot{e}_i(0)\right]t^4+\\ \left[\frac{84}{T^5}e_i(0)+\frac{45}{T^4}\dot{e}_i(0)+\frac{10}{T^3}\ddot{e}_i(0)+\frac{1}{T^2}\dddot{e}_i(0)\right]t^5-\\ \left[\frac{70}{T^6}e_i(0)+\frac{36}{T^5}\dot{e}_i(0)+\frac{15}{2T^4}\ddot{e}_i(0)+\frac{2}{3T^3}\dddot{e}_i(0)\right]t^6+\\ \left[\frac{20}{T^7}e_i(0)+\frac{10}{T^6}\dot{e}_i(0)+\frac{2}{T^5}\ddot{e}_i(0)+\frac{1}{6T^4}\dddot{e}_i(0)\right]t^7\\ \qquad \text{if} \quad 0\leqslant t\leqslant T\\ 0 \qquad \text{if} \quad t>T\end{cases}$$

位置指令分别为 $x_{11d}=\sin(\pi t/2)$，$x_{21d}=\cos(\pi t)$，$x_{31d}=1$。根据式(9.9)，控制器为

$$\boldsymbol{u}(t)=-\begin{bmatrix}u_1\\ u_2\\ u_3\end{bmatrix}-\frac{\boldsymbol{\sigma}}{\|\boldsymbol{\sigma}\|}[F(\boldsymbol{X},t)+K]$$

其中：

$$u_1=f(1)-\dddot{x}_{11d}-\dddot{p}_1(t)+4[x_{12}-\dot{x}_{11d}-\dot{p}_1(t)]+4[x_{13}-\ddot{x}_{11d}-\ddot{p}_1(t)]$$
$$u_2=f(2)-\dddot{x}_{21d}-\dddot{p}_2(t)+4[x_{22}-\dot{x}_{21d}-\dot{p}_2(t)]+4[x_{23}-\ddot{x}_{21d}-\ddot{p}_2(t)]$$
$$u_3=f(3)-\dddot{x}_{31d}-\dddot{p}_3(t)+4[x_{32}-\dot{x}_{31d}-\dot{p}_3(t)]+4[x_{33}-\ddot{x}_{31d}-\ddot{p}_3(t)]$$

系统的初始状态为 $\boldsymbol{x}_0=[1\ 0\ 0\ \ 0.5\ 0\ 0\ \ 0\ 0\ 0]$，$F(\boldsymbol{X},t)=2+e^{-t}$，$K=10$，$T=0.5$。采用连续函数代替切换函数，$\delta_0=0.03$，$\delta_1=5$。仿真结果如图 9-5～图 9-10 所示，三个输出的跟踪误差在 $T=0.5$ 时皆收敛为零，并有效地消除了抖振。

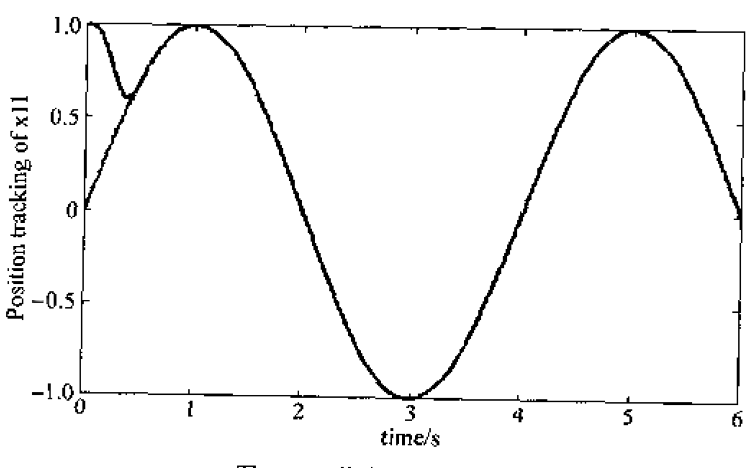


图 9-5 状态 x_{11} 的跟踪

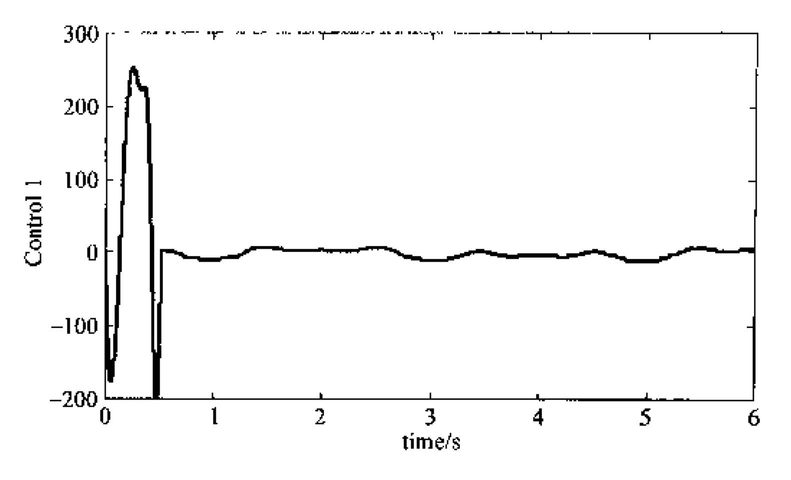


图 9-6　控制输入 u_1

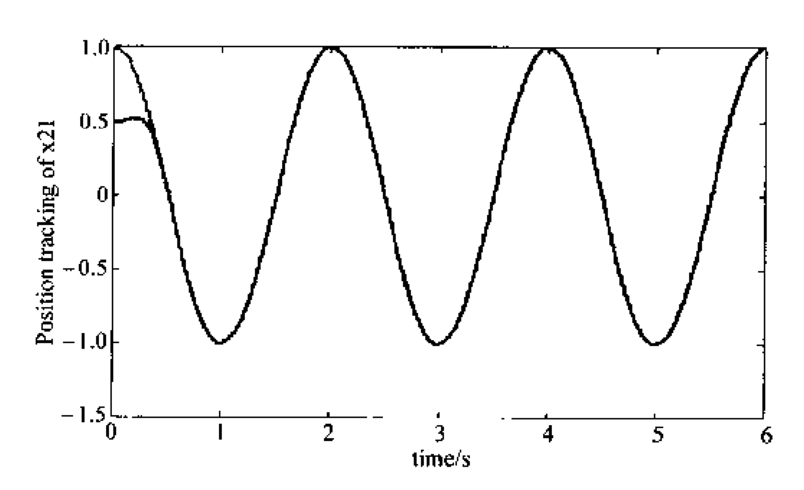


图 9-7　状态 x_{21} 的跟踪

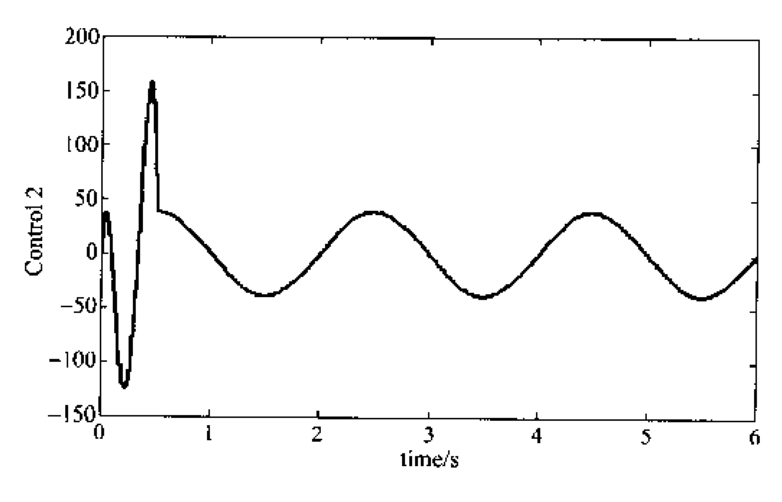


图 9-8　控制输入 u_2

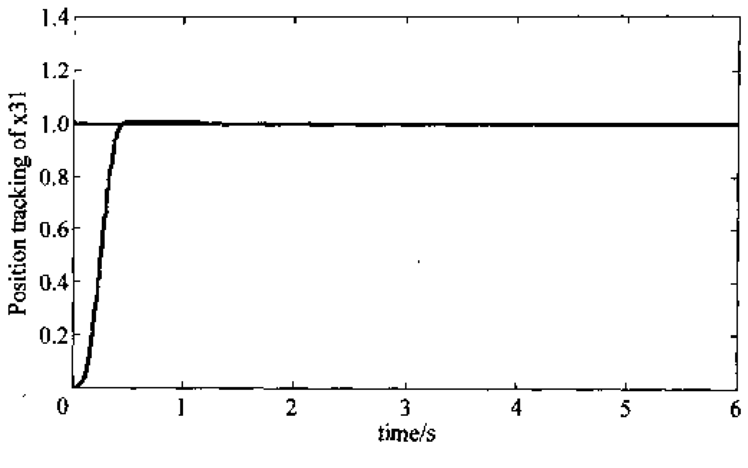


图 9-9 状态 x_{31} 的跟踪

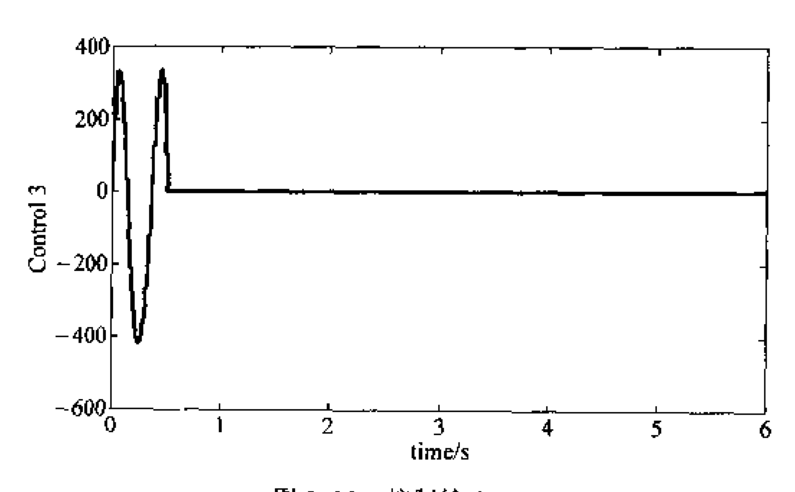


图 9-10 控制输入 u_3

仿真程序：

(1) 主程序(如图 9-11 所示)：chap9_2sim. mdl

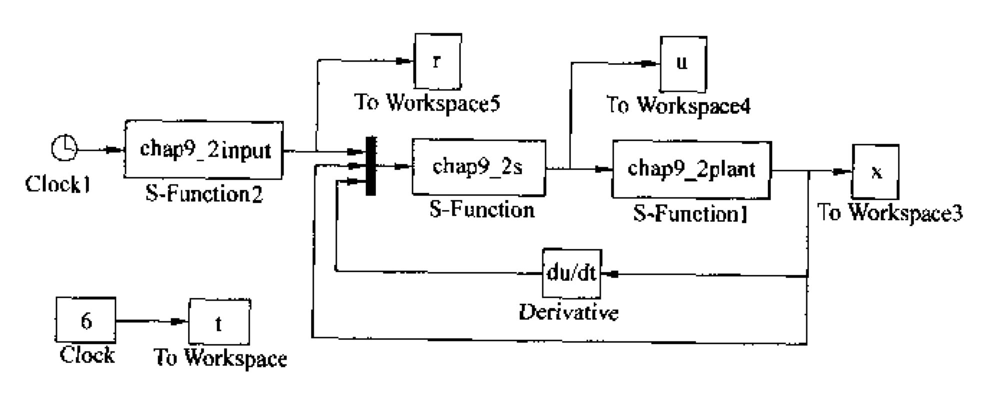


图 9-11 主程序图

(2) S 函数位置指令子程序：chap9_2input. m

```
%S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
```

```
 % Initialization
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
 % Outputs
  case 3,
    sys=mdlOutputs(t,x,u);
 % Unhandled flags
  case {2, 4, 9 }
    sys = [];
 % Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

 % mdlInitializeSizes
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys=simsizes(sizes);
x0=[];
str=[];
ts=[];
function sys=mdlOutputs(t,x,u)

x11d=sin(pi*t/2);
x21d=cos(pi*t);
x31d=1;

sys(1)=x11d;
sys(2)=x21d;
sys(3)=x31d;
```

(3) S 函数控制子程序：chap9_2s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
```

```
sizes.NumContStates   = 0;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 3;
sizes.NumInputs       = 21;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 1;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0 0];

function sys = mdlOutputs(t,x,u)
persistent e10 de10 dde10 ddde10 e20 de20 dde20 ddde20 e30 de30 dde30 ddde30
T=0.5;

x11d=u(1);
dx11d=pi/2*cos(pi*t/2);
ddx11d= -(pi/2)^2*sin(pi*t/2);
dddx11d= -(pi/2)^3*cos(pi*t/2);

x21d=u(2);
dx21d= -pi*sin(pi*t);
ddx21d= -pi^2*cos(pi*t);
dddx21d=pi^3*sin(pi*t);

x31d=u(3);
dx31d=0;
ddx31d=0;
dddx31d=0;

x=u(4:1:12);
dx=u(13:1:21);
if t==0
   e10=x(1);
   de10=x(2)-pi/2;
   dde10=x(3);
   ddde10=dx(3)+(pi/2)^3;

   e20=x(4)-1;
   de20=x(5);
   dde20=x(6)+pi^2;
   ddde20=dx(6);

   e30=x(7)-1;
   de30=x(8);
   dde30=x(9);
   ddde30=dx(9);
end

C=[4 0 0 4 0 0 1 0 0;
   0 4 0 0 4 0 0 1 0;
```

```
    0 0 4 0 0 4 0 0 1];

e1 = x(1) - x11d;
de1 = x(2) - dx11d;
dde1 = x(3) - ddx11d;
ddde1 = dx(3) - dddx11d;

e2 = x(4) - x21d;
de2 = x(5) - dx21d;
dde2 = x(6) - ddx21d;
ddde2 = dx(6) - dddx21d;

e3 = x(7) - x31d;
de3 = x(8) - dx31d;
dde3 = x(9) - ddx31d;
ddde3 = dx(9) - dddx31d;

e = [e1;e2;e3];
if t< = T
   A10 = 35/T^4 * e10 + 20/T^3 * de10 + 5/T^2 * dde10 + 2/3 * 1/T * ddde10;
   A11 = 84/T^5 * e10 + 45/T^4 * de10 + 10/T^3 * dde10 + 1/T^2 * ddde10;
   A12 = 70/T^6 * e10 + 36/T^5 * de10 + 15/(2 * T^4) * dde10 + 2/(3 * T^3) * ddde10;
   A13 = 20/T^7 * e10 + 10/T^6 * de10 + 2/(T^5) * dde10 + 1/(6 * T^4) * ddde10;

p1 = e10 + de10 * t + 1/2 * dde10 * t^2 + 1/6 * ddde10 * t^3 - A10 * t^4 + A11 * t^5 - A12 * t^6 + A13 * t^7;
   dp1 = de10 + dde10 * t + 1/2 * ddde10 * t^2 - A10 * 4 * t^3 + A11 * 5 * t^4 - A12 * 6 * t^5 + A13 * 7 * t^6;
   ddp1 = dde10 + ddde10 * t - A10 * 4 * 3 * t^2 + A11 * 5 * 4 * t^3 - A12 * 6 * 5 * t^4 + A13 * 7 * 6 * t^5;
   dddp1 = ddde10 - A10 * 4 * 3 * 2 * t + A11 * 5 * 4 * 3 * t^2 - A12 * 6 * 5 * 4 * t^3 + A13 * 7 * 6 * 5 * t^4;

   A20 = 35/T^4 * e20 + 20/T^3 * de20 + 5/T^2 * dde20 + 2/3 * 1/T * ddde20;
   A21 = 84/T^5 * e20 + 45/T^4 * de20 + 10/T^3 * dde20 + 1/T^2 * ddde20;
   A22 = 70/T^6 * e20 + 36/T^5 * de20 + 15/(2 * T^4) * dde20 + 2/(3 * T^3) * ddde20;
   A23 = 20/T^7 * e20 + 10/T^6 * de20 + 2/(T^5) * dde20 + 1/(6 * T^4) * ddde20;

p2 = e20 + de20 * t + 1/2 * dde20 * t^2 + 1/6 * ddde20 * t^3 - A20 * t^4 + A21 * t^5 - A22 * t^6 + A23 * t^7;
   dp2 = de20 + dde20 * t + 1/2 * ddde20 * t^2 - A20 * 4 * t^3 + A21 * 5 * t^4 - A22 * 6 * t^5 + A23 * 7 * t^6;
   ddp2 = dde20 + ddde20 * t - A20 * 4 * 3 * t^2 + A21 * 5 * 4 * t^3 - A22 * 6 * 5 * t^4 + A23 * 7 * 6 * t^5;
   dddp2 - ddde20 - A20 * 4 * 3 * 2 * t + A21 * 5 * 4 * 3 * t^2 - A22 * 6 * 5 * 4 * t^3 + A23 * 7 * 6 * 5 * t^4;

   A30 = 35/T^4 * e30 + 20/T^3 * de30 + 5/T^2 * dde30 + 2/3 * 1/T * ddde30;
   A31 = 84/T^5 * e30 + 45/T^4 * de30 + 10/T^3 * dde30 + 1/T^2 * ddde30;
   A32 = 70/T^6 * e30 + 36/T^5 * de30 + 15/(2 * T^4) * dde30 + 2/(3 * T^3) * ddde30;
   A33 = (20/T^7 * e30 + 10/T^6 * de30 + 2/(T^5) * dde30 + 1/(6 * T^4) * ddde30);

p3 = e30 + de30 * t + 1/2 * dde30 * t^2 + 1/6 * ddde30 * t^3 - A30 * t^4 + A31 * t^5 - A32 * t^6 + A33 * t^7;
   dp3 = de30 + dde30 * t + 1/2 * ddde30 * t^2 - A30 * 4 * t^3 + A31 * 5 * t^4 - A32 * 6 * t^5 + A33 * 7 * t^6;
   ddp3 = dde30 + ddde30 * t - A30 * 4 * 3 * t^2 + A31 * 5 * 4 * t^3 - A32 * 6 * 5 * t^4 + A33 * 7 * 6 * t^5;
   dddp3 = ddde30 - A30 * 4 * 3 * 2 * t + A31 * 5 * 4 * 3 * t^2 - A32 * 6 * 5 * 4 * t^3 + A33 * 7 * 6 * 5 * t^4;
else
   p1 = 0;p2 = 0;p3 = 0;
   dp1 = 0;dp2 = 0;dp3 = 0;
```

```
    ddp1 = 0;ddp2 = 0;ddp3 = 0;
    dddp1 = 0;dddp2 = 0;dddp3 = 0;
end

f = [ - 1.5 * x(3)^2 * cos(3 * x(4)),3 * x(5) * sin(x(7)),x(9) * cos(x(1)) * sin(x(4))];
u1 = f(1) - dddx11d - dddp1 + 4 * (x(2) - dx11d - dp1) + 4 * (x(3) - ddx11d - ddp1);
u2 = f(2) - dddx21d - dddp2 + 4 * (x(5) - dx21d - dp2) + 4 * (x(6) - ddx21d - ddp2);
u3 = f(3) - dddx31d - dddp3 + 4 * (x(8) - dx31d - dp3) + 4 * (x(9) - ddx31d - ddp3);

rou1 = dde1 + 4 * de1 + 4 * e1 - ddp1 - 4 * dp1 - 4 * p1;
rou2 = dde2 + 4 * de2 + 4 * e2 - ddp2 - 4 * dp2 - 4 * p2;
rou3 = dde3 + 4 * de3 + 4 * e3 - ddp3 - 4 * dp3 - 4 * p3;
rou = [rou1;rou2;rou3];

delta0 = 0.03;
delta1 = 5;
delta = delta0 + delta1 * norm(e);
mrou = norm(rou) + delta;

K = 10;
F = 2 + exp( - t);
ut = - [u1;u2;u3] - rou/mrou * (F + K);

sys(1) = ut(1);
sys(2) = ut(2);
sys(3) = ut(3);
```

(4) S 函数被控对象子程序:chap9_2plant. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 9;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 9;
sizes.NumInputs       = 3;
```

```
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 1;
sys  = simsizes(sizes);
x0   = [1,0,0,0.5,0,0,0,0,0];
str  = [];
ts   = [0 0];

function sys = mdlDerivatives(t,x,u)
f = [-1.5 * x(3)^2 * cos(3 * x(4)),3 * x(5) * sin(x(7)),x(9) * cos(x(1)) * sin(x(4))];
b = [1 0 0;0 1 0;0 0 1];
df = [sin(t) * sin(x(5)^2) * cos(3 * x(1)),exp(-t) * cos(x(3)) * sin(x(7)),cos(t) * sin(x
(6))];

sys(1) = x(2);
sys(2) = x(3);
sys(3) = f(1) + df(1) + u(1);

sys(4) = x(5);
sys(5) = x(6);
sys(6) = f(2) + df(2) + u(2);

sys(7) = x(8);
sys(8) = x(9);
sys(9) = f(3) + df(3) + u(3);

function sys = mdlOutputs(t,x,u)

sys(1) = x(1);
sys(2) = x(2);
sys(3) = x(3);
sys(4) = x(4);
sys(5) = x(5);
sys(6) = x(6);
sys(7) = x(7);
sys(8) = x(8);
sys(9) = x(9);
```

(5) 作图子程序:chap9_2plot.m

```
close all;

figure(1);
plot(t,r(:,1),'r',t,x(:,1),'b');
xlabel('time(s)');ylabel('Position tracking of x11');
figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('control 1');

figure(3);
plot(t,r(:,2),'r',t,x(:,4),'b');
xlabel('time(s)');ylabel('Position tracking of x21');
```

```
figure(4);
plot(t,u(:,2),'r');
xlabel('time(s)');ylabel('control 2');

figure(5);
plot(t,r(:,3),'r',t,x(:,7),'b');
xlabel('time(s)');ylabel('Position tracking of x31');
figure(6);
plot(t,u(:,3),'r');
xlabel('time(s)');ylabel('control 3');
```

9.2 动态 Terminal 滑模控制

9.2.1 系统描述

针对如下 SISO 系统：

$$\begin{cases}\dot{x}_i = x_{i+1} \\ \dot{x}_n = f(\boldsymbol{x},t) + b(\boldsymbol{x},t)u + d(\boldsymbol{x},t)\end{cases} \tag{9.14}$$

其中 $i=1,2,\cdots,n-1$，$\boldsymbol{x}=[x_1 x_2 \cdots x_n]^{\mathrm{T}}=[x_1 \dot{x}_1 \cdots x_1^{(n-1)}]^{\mathrm{T}}$。$f(\boldsymbol{x},t)$和 $b(\boldsymbol{x},t)$为未知连续函数，$b(\boldsymbol{x},t)>0$；$d(\boldsymbol{x},t)$为外部干扰，$|d(\boldsymbol{x},t)|\leqslant D_{\max}$，并假设外部干扰为慢时变，即 $\dot{d}(x,t)=0$。

9.2.2 动态 Terminal 滑模控制器的设计

1. Terminal 滑模面的设计

定义跟踪误差为

$$\boldsymbol{E} = \boldsymbol{x} - \boldsymbol{x}_d = [e\dot{e}\cdots e^{(n)}]^{\mathrm{T}} \tag{9.15}$$

其中 $e=x_1-x_{1d}$。

定义 Terminal 切换函数为

$$s(\boldsymbol{x},t) = \boldsymbol{CE} - \boldsymbol{W}(t) \tag{9.16}$$

其中 $\boldsymbol{C}=[c_1 c_2 \cdots c_n]$，$c_i(i=1,2,\cdots,n)$为正常数且 $c_n=1$。

定义

$$\boldsymbol{W}(t) = \boldsymbol{CP}(t) \tag{9.17}$$

其中 $\boldsymbol{P}(t)=[p(t)\dot{p}(t)\cdots p^{(n)}(t)]^{\mathrm{T}}$。

定义动态 Terminal 滑动模面为

$$\sigma(\boldsymbol{x},t) = \dot{s}(\boldsymbol{x},t) + \lambda s(\boldsymbol{x},t) \tag{9.18}$$

其中 λ 为正的常数。

假设 1 设计 Terminal 函数 $p(t):\mathbf{R}_+\rightarrow\mathbf{R}$，$p(t)\in C^n[0,\infty)$，$\dot{p},\ddot{p},\cdots,p^{(n)}\in\mathbf{L}^{\infty}$。对于某一个 $T>0$，$p(t)$在$[0,T]$内有界，且 $p(0)=e(0)$，$\dot{p}(0)=\dot{e}(0)$，$\cdots$，$p^{(n)}(0)=e^{(n)}(0)$。其中 $C^n[0,\infty)$表示定义在$[0,\infty)$的所有 n 阶可微的连续函数。

选取函数 $p(t)$ 为

$$p(t)=\begin{cases}\sum\limits_{k=0}^{n}\dfrac{1}{k!}e(0)^{(k)}t^{k}+\sum\limits_{j=0}^{n}\left[\sum\limits_{l=0}^{n}\dfrac{a_{jl}}{T^{j-l+n+1}}e(0)^{(l)}\right]t^{j+n+1} & \text{if}\quad 0\leqslant t\leqslant T\\ 0 & \text{if}\quad t>T\end{cases}\tag{9.19}$$

其中 a_{jl} 可通过假设 1 得到。

以 $n=2$ 为例，Terminal 函数 $p(t)$ 表示为

$$p(t)=\begin{cases}e_0+\dot{e}_0t+\dfrac{1}{2}\ddot{e}_0t^2-\left(\dfrac{a_{00}}{T^3}e_0+\dfrac{a_{01}}{T^2}\dot{e}_0+\dfrac{a_{02}}{T}\ddot{e}_0\right)t^3+\\ \left(\dfrac{a_{10}}{T^4}e_0+\dfrac{a_{11}}{T^3}\dot{e}_0+\dfrac{a_{12}}{T^2}\ddot{e}_0\right)t^4-\left(\dfrac{a_{20}}{T^5}e_0+\dfrac{a_{21}}{T^4}\dot{e}_0+\dfrac{a_{22}}{T^3}\ddot{e}_0\right)t^5 & \text{if } 0\leqslant t\leqslant T\\ 0 & \text{if}\quad t>T\end{cases}\tag{9.20}$$

根据假设 1，$p(t)$ 在 $t=T$ 时为连续可导函数，Terminal 函数 $p(t)$ 及其导数 $\dot{p}(t)$，$\ddot{p}(t)$ 在 $t=T$ 时为零。可得如下 3 组方程组：

$$\begin{cases}a_{00}+a_{10}+a_{20}=-1\\ 3a_{00}+4a_{10}+5a_{20}=0\\ 6a_{00}+12a_{10}+20a_{20}=0\end{cases}\tag{9.21}$$

$$\begin{cases}a_{01}+a_{11}+a_{21}=-1\\ 3a_{01}+4a_{11}+5a_{21}=-1\\ 6a_{01}+12a_{11}+20a_{21}=0\end{cases}\tag{9.22}$$

$$\begin{cases}a_{02}+a_{12}+a_{22}=-\dfrac{1}{2}\\ 3a_{02}+4a_{12}+5a_{22}=-1\\ 6a_{02}+12a_{12}+20a_{22}=-1\end{cases}\tag{9.23}$$

解上述方程，可得 a_{jl}：

$$\begin{cases}a_{00}=-10\\ a_{10}=15\\ a_{20}=-6\end{cases}\quad\begin{cases}a_{01}=-6\\ a_{11}=8\\ a_{21}=-3\end{cases}\quad\begin{cases}a_{02}=-1.5\\ a_{12}=1.5\\ a_{22}=-0.5\end{cases}\tag{9.24}$$

2. 动态 Terminal 滑模控制器的设计

设计动态滑模控制律为

$$\dot{u}(t)=-\frac{1}{c_nb(\boldsymbol{x},t)}[I_1u+I_2+I_3+(c_{n-1}+\lambda c_n)(D+\eta)\operatorname{sgn}(\sigma)]\tag{9.25}$$

其中 $\eta>0$，I_1，I_2 和 I_3 见式(9.36)～式(9.38)，则 $\sigma(\boldsymbol{x},t)$ 将在有限时间 T 趋近于零。

稳定性分析：

定义 Lyapunov 函数

$$V(t)=\frac{1}{2}\sigma^{\mathrm{T}}\sigma\tag{9.26}$$

由于

$$s(\boldsymbol{x},t)=\boldsymbol{C}(\boldsymbol{E}-\boldsymbol{P})=\sum_{i=1}^{n}c_i(e^{(i-1)}-p^{(i-1)})\tag{9.27}$$

则

$$\dot{s}(\boldsymbol{x},t)=\boldsymbol{C}\dot{\boldsymbol{E}}-\boldsymbol{C}\dot{\boldsymbol{P}}$$
$$=\boldsymbol{C}\cdot[\dot{e}\ \ddot{e}\cdots e^{(n)}]^{\mathrm{T}}-\boldsymbol{C}\cdot[\dot{p}\ \ddot{p}\cdots p^{(n)}] \tag{9.28}$$
$$=c_n[e^{(n)}-p^{(n)}]+\sum_{k=1}^{n-1}c_k[e^{(k)}-p^{(k)}]$$

$$\sigma=\dot{s}+\lambda s=c_n(e^{(n)}-p^{(n)})+\sum_{k=1}^{n-1}c_k[e^{(k)}-p^{(k)}]+\lambda\sum_{l=1}^{n}c_l[e^{(l-1)}-p^{(l-1)}] \tag{9.29}$$

$$\dot{\sigma}=c_n(e^{(n+1)}-p^{(n+1)})+\sum_{k=1}^{n-1}c_k[e^{(k+1)}-p^{(k+1)}]+\lambda\sum_{l=1}^{n}c_l[e^{(l)}-p^{(l)}] \tag{9.30}$$

由于

$$e^{(n+1)}=\dot{e}^{(n)}=\dot{x}_1^{(n)}-\dot{x}_{1d}^{(n)}=\ddot{x}_n-\ddot{x}_{nd} \tag{9.31}$$

$$c_n(e^{(n+1)}-p^{(n+1)})=c_n(\ddot{x}_n-\ddot{x}_{nd}-p^{(n+1)})$$
$$=c_n(\dot{f}(\boldsymbol{x},t)+\dot{b}(\boldsymbol{x},t)u+b(\boldsymbol{x},t)\dot{u}+\dot{d}(\boldsymbol{x},t)-\ddot{x}_{nd}^{(T)}-p^{(n+1)}) \tag{9.32}$$

$$\sum_{k=1}^{n-1}c_k(e^{(k+1)}-p^{(k+1)})=c_{n-1}(e^{(n)}-p^{(n)})+\sum_{k=1}^{n-2}c_k(e^{(k+1)}-p^{(k+1)})$$
$$=c_{n-1}[f(\boldsymbol{x},t)+b(\boldsymbol{x},t)u+d(\boldsymbol{x},t)-\dot{x}_{nd}-p^{(n)}]+\sum_{k=1}^{n-2}c_k(e^{(k+1)}-p^{(k+1)}) \tag{9.33}$$

$$\lambda\sum_{l=1}^{n}c_l[e^{(l)}-p^{(l)}]=\lambda c_n(e^{(n)}-p^{(n)})+\lambda\sum_{l=1}^{n-1}c_l[e^{(l)}-p^{(l)}]$$
$$=\lambda c_n[f(\boldsymbol{x},t)+b(\boldsymbol{x},t)u+d(\boldsymbol{x},t)-\dot{x}_{nd}-p^{(n)}]+\lambda\sum_{l=1}^{n-1}c_l[e^{(l)}-p^{(l)}] \tag{9.34}$$

则

$$\dot{\sigma}=[c_n\dot{b}(\boldsymbol{x},t)+(c_{n-1}+\lambda c_n)b(\boldsymbol{x},t)]u+(c_{n-1}+\lambda c_n)d(\boldsymbol{x},t)$$
$$+c_n[b(\boldsymbol{x},t)\dot{u}+\dot{d}(\boldsymbol{x},t)]+c_n[\dot{f}(\boldsymbol{x},t)-\ddot{x}_{nd}-p_{n+1}]$$
$$+(c_{n-1}+\lambda c_n)[f(\boldsymbol{x},t)-\dot{x}_{nd}-p^{(n)}]+\sum_{k=1}^{n-2}c_k(e^{(k+1)}-p^{(k+1)}) \tag{9.35}$$
$$+\lambda\sum_{l=1}^{n-1}c_l(e^{(l)}-p^{(l)})$$
$$=I_1u+(c_{n-1}+\lambda c_n)d(\boldsymbol{x},t)+c_nb(\boldsymbol{x},t)\dot{u}+I_2+I_3$$

其中

$$I_1=c_n\dot{b}(x,t)+(c_{n-1}+\lambda c_n)b(\boldsymbol{x},t) \tag{9.36}$$

$$I_2=c_n[\dot{f}(\boldsymbol{x},t)-\ddot{x}_{nd}-p_{n+1}]+(c_{n-1}+\lambda c_n)[f(\boldsymbol{x},t)-\dot{x}_{nd}-p^{(n)}] \tag{9.37}$$

$$I_3=\sum_{k=1}^{n-2}c_k[e^{(k+1)}-p^{(k+1)}]+\lambda\sum_{l=1}^{n-1}c_l[e^{(l)}-p^{(l)}] \tag{9.38}$$

将控制律式(9.25)代入式(9.35),得

$$\dot{\sigma}=(c_{n-1}+\lambda c_n)[d(x,t)-(D+\eta)\mathrm{sgn}(\sigma)] \tag{9.39}$$

$$\sigma\dot{\sigma}=(c_{n-1}+\lambda c_n)[d(\boldsymbol{x},t)\sigma-(D+\eta)\,|\sigma|]\leqslant-(c_{n-1}+\lambda c_n)\eta\,|\sigma| \tag{9.40}$$

如果$|\sigma|\neq0$,则 $\dot{V}<0$。

结论 1:由假设 1 可知

$$\begin{aligned} s(\boldsymbol{x},0) &= \boldsymbol{CE}(0) - \boldsymbol{W}(0) = \boldsymbol{C}(\boldsymbol{E}(0) - \boldsymbol{P}(0)) \\ &= \boldsymbol{C} \cdot \{[e(0)\cdots e^{(n)}(0)]^{\mathrm{T}} - [p(0)\cdots p^{(n)}(0)]^{\mathrm{T}}\} \\ &= 0 \end{aligned} \tag{9.41}$$

则可知系统的初始状态已经在滑模面上，消除了滑模的到达阶段，确保闭环系统的全局鲁棒性和稳定性。

结论 2：考虑到 $\sigma(\boldsymbol{x},0)\to 0$ 时，由式(9.18)可知，如果 λ 值取得大些，可保证在所有时间内 $s(\boldsymbol{x},t)\to 0$。

结论 3：取 $\boldsymbol{\xi}(t)=\boldsymbol{E}(t)-\boldsymbol{P}(t)$，则

$$s(\boldsymbol{x},t) = \boldsymbol{CE}(t) - \boldsymbol{W}(t) = \boldsymbol{CE}(t) - \boldsymbol{CP}(t) = \boldsymbol{C\xi}(t) \tag{9.42}$$

可见，如果选择了合适的 $\boldsymbol{P}(t)=0, \forall t\geqslant T$，则跟踪误差在有限时间 T 内收敛为零。

9.2.3 仿真实例

考虑如下被控对象：

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(\boldsymbol{x},t) + b(\boldsymbol{x},t)u(t) + d(\boldsymbol{x},t) \end{cases} \tag{9.43}$$

其中 $\boldsymbol{x}=[x_1 x_2]=[x_1 \dot{x}_1]$。

假设

$$f(\boldsymbol{x},t) = [10 + 0.01\sin(t)]x_1, \dot{f}(\boldsymbol{x},t) = 0.01\cos(t)x_1 + [10 + 0.01\sin(t)]x_2$$

$$b(\boldsymbol{x},t) = 130 + 0.1\cos(t), \dot{b}(\boldsymbol{x},t) = -0.1\sin(t)$$

$$d(\boldsymbol{x},t) = 3.9 + 0.1\sin(t/2), D_{\max} = 4.0$$

由于 $n=2$，则 $p(t)$ 按式(9.20)进行设计，$s(\boldsymbol{x},t)$ 和 $\sigma(\boldsymbol{x},t)$ 按式(9.16)和式(9.18)设计。

取 $c_1=5, c_2=1, \lambda=15, \eta=1.50$，位置指令取 $x_d(t)=0.50\sin(t)$，系统初始状态为 $\boldsymbol{x}=[0.6\ \ 0.0]$，Terminal 时间取 $T=0.80$s，仿真时间为 30，仿真结果如图 9-12～图 9-15 所示。由仿真结果可见，采用动态滑模控制方法，大大降低了抖振。

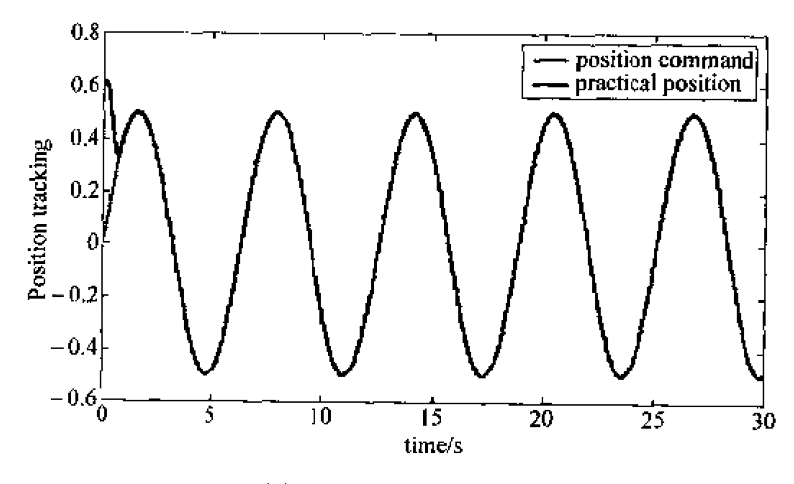


图 9-12　正弦位置跟踪

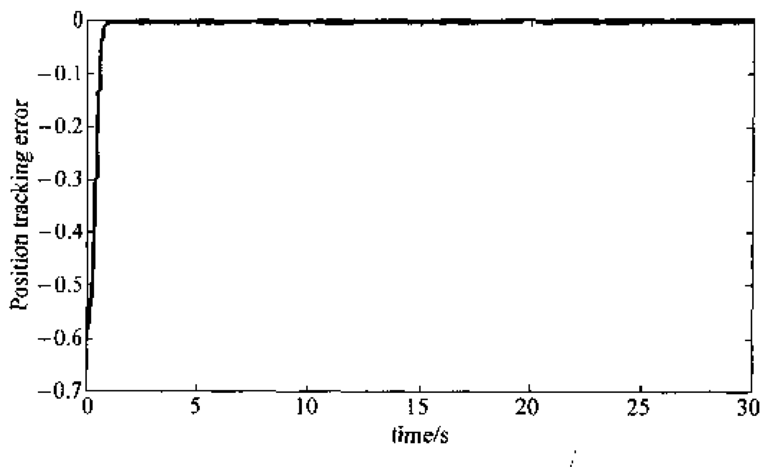


图 9-13 正弦位置跟踪误差

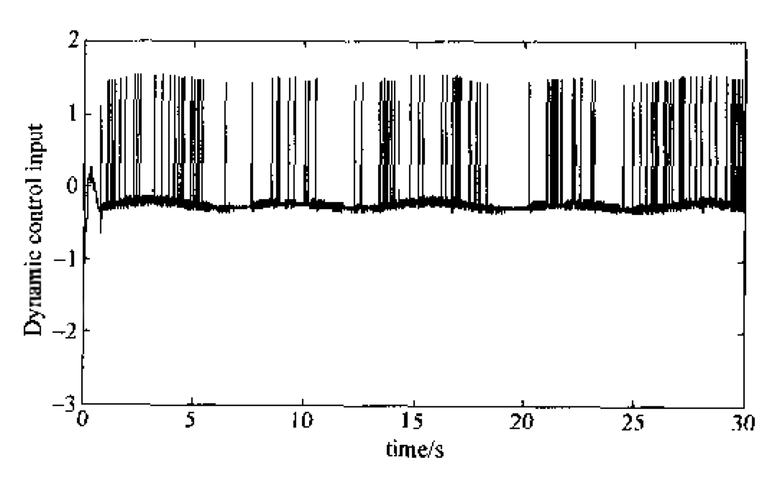


图 9-14 动态控制输入 $\dot{u}$

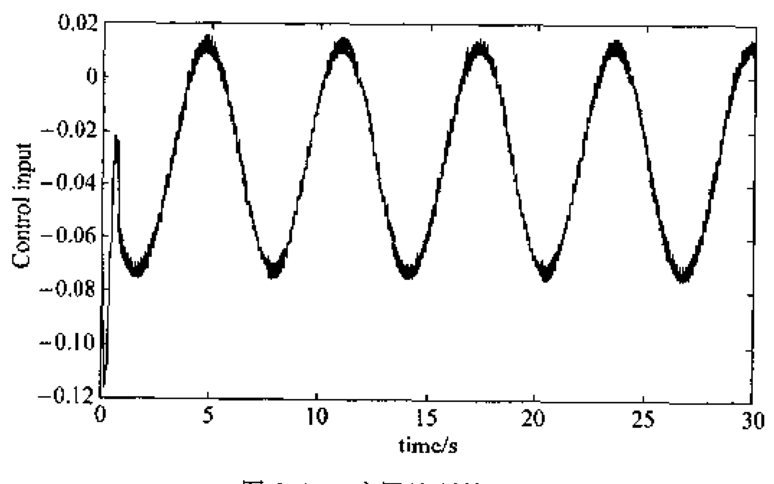


图 9-15 实际控制输入 u

仿真程序：

(1) Simulink 主程序(如图 9-16 所示)：chap9_3sim. mdl

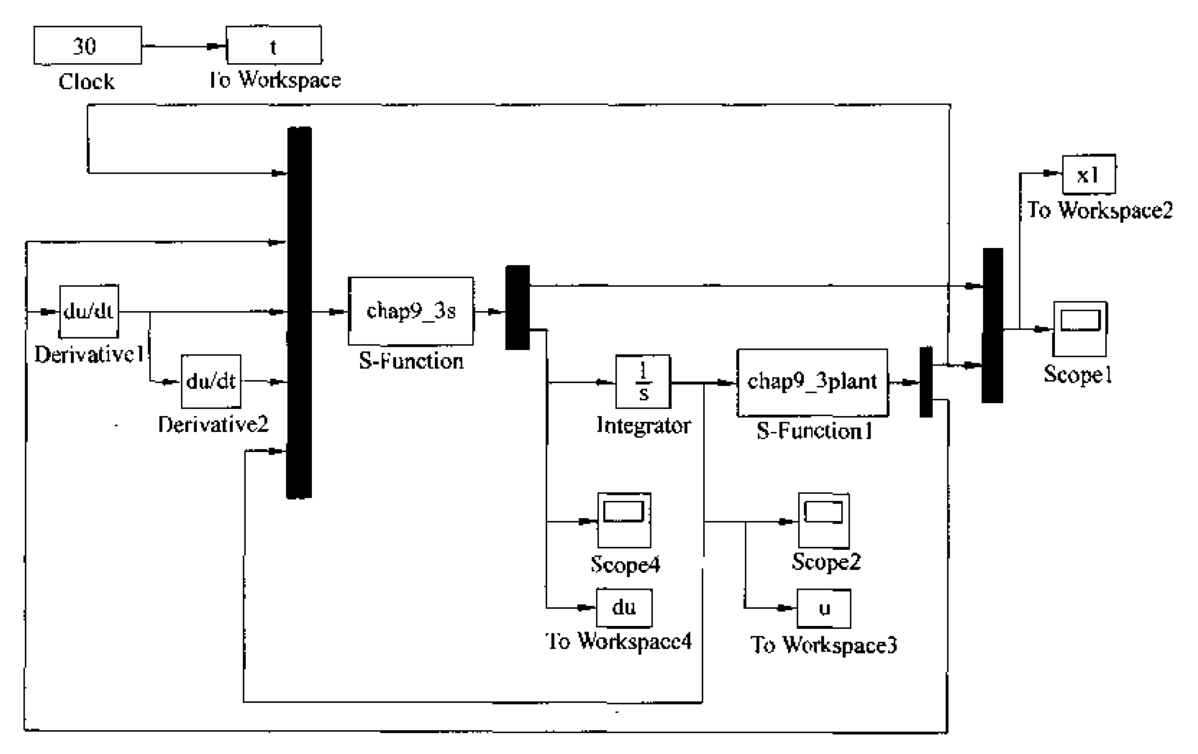


图 9-16　主程序图

(2) 控制器 S 函数:chap9_3s. m

```
function [sys,x0,str,ts] = controller(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9},
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 0;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 5;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 0;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [];
```

```
function sys = mdlOutputs(t,x,u)
persistent e0 de0 dde0 ddde0

x1 = u(1);
dx1 = u(2);
ddx1 = u(3);
dddx1 = u(4);
ut = u(5);

r = 0.5 * sin(t);
dr = 0.5 * cos(t);
ddr = - 0.5 * sin(t);
dddr = - 0.5 * cos(t);
ddddr = 0.5 * sin(t);

if t == 0
   e0 = x1 - r;de0 = dx1 - dr;dde0 = ddx1 - ddr;ddde0 = dddx1 - dddr;
end

   fx = (10 + 0.01 * sin(t)) * x1;
   bx = 130 + 0.1 * cos(t);

   dfx = 0.01 * cos(t) * x1 + (10 + 0.01 * sin(t)) * dx1;
   dbx = - 0.1 * sin(t);

e = x1 - r;
de = dx1 - dr;
dde = ddx1 - ddr;
ddde = dddx1 - dddr;

T = 0.8;    % Set terminal time
if t<= T
   P1 = 10/T^3 * e0 + 6/T^2 * de0 + 1.5/T * dde0;
   P2 = 15/T^4 * e0 + 8/T^3 * de0 + 1.5/T^2 * dde0;
   P3 = 6/T^5 * e0 + 3/T^4 * de0 + 0.5/T^3 * dde0;

   Pt = e0 + de0 * t + 1/2 * dde0 * t^2 - P1 * t^3 + P2 * t^4 - P3 * t^5;
   dPt = de0 + dde0 * t - P1 * 3 * t^2 + P2 * 4 * t^3 - P3 * 5 * t^4;
   ddPt = dde0 - P1 * 3 * 2 * t + P2 * 4 * 3 * t^2 - P3 * 5 * 4 * t^3;
   dddPt = ddde0 - P1 * 3 * 2 + P2 * 4 * 3 * 2 * t - P3 * 5 * 4 * 3 * t^2;
else
   Pt = 0;dPt = 0;ddPt = 0;dddPt = 0;
end

nmn = 15;
c1 = 5;
c2 = 1;
s = c1 * (e - Pt) + c2 * (de - dPt);
ds = c1 * (de - dPt) + c2 * (dde - ddPt);
rou = ds + nmn * s;
```

```
xite = 1.5;
Dmax = 4.0;

I1 = c2 * dbx + (c1 + nmn * c2) * bx;
I2 = c2 * (dfx - dddr - dddPt) + (c1 + nmn * c2) * (fx - ddr - ddPt);
I3 = nmn * c1 * (de - dPt);

dut = - 1/bx * (I1 * ut + I2 + I3 + (c1 + nmn * c2) * (Dmax + xite) * sign(rou));

sys(1) = r;
sys(2) = dut;
```

(3) 被控对象 S 函数:chap9_3plant.m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
  case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 },
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
  end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 2;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 1;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 0;

sys = simsizes(sizes);
x0 = [0.6,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
```

```
    fx = (10 + 0.01 * sin(t)) * x(1);
    bx = 130 + 0.1 * cos(t);

    dfx = 0.01 * cos(t) * x(1) + (10 + 0.01 * sin(t)) * x(2);
    dbx = - 0.1 * sin(t);
    dt = 3.9 + 0.1 * sin(t/2);
sys(1) = x(2);
sys(2) = fx + bx * u + dt;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(4) 作图程序:chap9_3plot.m

```
close all;

figure(1);
plot(t,x1(:,1),'r',t,x1(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,x1(:,1) - x1(:,2),'r');
xlabel('time(s)');ylabel('Position tracking error');

figure(3);
plot(t,u,'r');
xlabel('time(s)');ylabel('Control input');

figure(4);
plot(t,du,'r');
xlabel('time(s)');ylabel('Dynamic control input');
```

9.3 快速 Terminal 滑模控制器的设计

快速 Terminal 滑模控制可使系统状态在有限时间内收敛为零，突破了普通滑模控制在线性滑模面条件下状态渐近收敛的特点，系统的动态性能优于普通滑模控制。并且相对于线性滑模控制，快速 Terminal 滑模控制无切换项，可有效地消除抖振。快速 Terminal 滑模控制的提出为滑动模态控制理论带来了新的发展方向。

9.3.1 快速 Terminal 滑模的设计

1. 传统快速 Terminal 滑动模态

一种传统的快速 Terminal 滑动模态的形式[1]为

$$s = \dot{x} + \beta x^{q/p} = 0 \tag{9.44}$$

其中 $x \in \mathbf{R}$ 为状态变量，$\beta > 0$，p，q（$p > q$）为正奇数。

由式(9.44)得

$$\frac{\mathrm{d}x}{\mathrm{d}t}=-\beta x^{q/p}$$

即

$$\mathrm{d}t=-\frac{1}{\beta}x^{-q/p}\mathrm{d}x$$

对上式进行定积分得

$$\int_0^t \mathrm{d}t=\int_{x_0}^0 -\frac{1}{\beta}x^{-q/p}\mathrm{d}x$$

从而得到从任意初始状态 $x(0)\neq 0$ 沿滑动模态式(9.44)到达平衡状态 $x=0$ 的时间为

$$t_s=\frac{p}{\beta(p-q)}\mid x(0)\mid^{(p-q)/p} \tag{9.45}$$

平衡状态 $x=0$ 也叫 Terminal 吸引子。由于引入了非线性部分 $\beta x^{q/p}$，改善了向平衡状态收敛的收敛速度，并且越远离平衡状态，收敛速度越快。然而，Terminal 滑模控制在收敛时间上却未必是最优的，主要原因在于，在系统状态接近平衡状态时，非线性滑动模态式(9.44)的收敛速度要比线性滑动模态($p=q$)的收敛速度慢。为此，文献[2]提出了一种新型全局快速 Terminal 滑动模态。

2. 全局快速 Terminal 滑动模态

综合考虑线性滑动模态与快速 Terminal 滑动模态，一种新型全局快速 Terminal 滑动模态为

$$s=\dot{x}+\alpha x+\beta x^{q/p}=0 \tag{9.46}$$

其中 $x\in\mathbf{R}$ 为状态变量，$\alpha,\beta>0$，p 和 $q(p>q)$ 为正奇数。

由式(9.46)得

$$x^{-q/p}\frac{\mathrm{d}x}{\mathrm{d}t}+\alpha x^{1-q/p}=-\beta \tag{9.47}$$

令 $y=x^{1-q/p}$，则 $\frac{\mathrm{d}y}{\mathrm{d}t}=\frac{p-q}{p}x^{-\frac{q}{p}}\frac{\mathrm{d}x}{\mathrm{d}t}$，式(9.47)写为

$$\frac{\mathrm{d}y}{\mathrm{d}t}-\frac{p-q}{p}\alpha y=-\frac{p-q}{p}\beta \tag{9.48}$$

由于一阶线性微分方程 $\frac{\mathrm{d}y}{\mathrm{d}x}+P(x)y=Q(x)$ 的通解为

$$y=\mathrm{e}^{-\int P(x)\mathrm{d}x}\left(\int Q(x)\mathrm{e}^{\int P(x)\mathrm{d}x}\mathrm{d}x+C\right)$$

则式(9.48)的解为

$$y=\mathrm{e}^{-\int_0^t\frac{p-q}{p}\alpha\mathrm{d}t}\left(\int_0^t -\frac{p-q}{p}\beta\mathrm{e}^{\int_0^t\frac{p-q}{p}\alpha\mathrm{d}t}\mathrm{d}t+C\right)=\mathrm{e}^{-\int_0^t\frac{p-q}{p}\alpha\mathrm{d}t}\left(\int_0^t -\frac{p-q}{p}\beta\mathrm{e}^{\frac{p-q}{p}\alpha t}\mathrm{d}t+C\right)$$

$t=0$ 时，$C=y(0)$，上式变为

$$y=\mathrm{e}^{-\frac{p-q}{p}\alpha t}\left(-\frac{p-q}{p}\beta\frac{p}{(p-q)\alpha}\mathrm{e}^{\frac{p-q}{p}\alpha t}\Big|_0^t+y(0)\right)=-\frac{\beta}{\alpha}+\frac{\beta}{\alpha}\mathrm{e}^{-\frac{p-q}{p}\alpha t}+y(0)\mathrm{e}^{-\frac{p-q}{p}\alpha t}$$

由于 $x=0$ 时，$y=0$，$t=t_s$，上式变为

$$\frac{\beta}{\alpha}\mathrm{e}^{\frac{p-q}{p}\alpha t_s}+y(0)\mathrm{e}^{\frac{p-q}{p}\alpha t_s}=\frac{\beta}{\alpha}$$

即

$$\left(\frac{\beta}{\alpha}+y(0)\right)\mathrm{e}^{-\frac{p-q}{p}\alpha t_s}=\frac{\beta}{\alpha}$$

$$\frac{[\beta+\alpha y(0)]}{\beta}=\mathrm{e}^{\frac{p-q}{p}\alpha t_s}$$

在滑动模态上从任意初始状态 $x(0)\neq 0$ 收敛到平衡状态 $x=0$ 的时间为

$$t_s=\frac{p}{\alpha(p-q)}\ln\frac{\alpha x(0)^{(p-q)/p}+\beta}{\beta} \tag{9.49}$$

通过设定 α,β,p,q 可使系统在有限时间 t_s 内到达平衡状态。由式(9.46),有

$$\dot{x}=-\alpha x-\beta x^{q/p} \tag{9.50}$$

当系统状态 x 远离零点时,收敛时间主要由快速 Terminal 吸引子即式 $\dot{x}=-\beta x^{q/p}$ 决定;而当系统状态 x 接近平衡状态 $x=0$ 时,收敛时间主要由式 $\dot{x}=-\alpha x$ 决定,x 呈指数快速衰减。因此,滑动模态式(9.46)既引入了 Terminal 吸引子,使得系统状态在有限时间收敛,又保留了线性滑动模态在接近平衡态时的快速性,从而实现系统状态快速、精确地收敛到平衡状态,所以称滑动模态式(9.46)为全局快速滑动模态。

全局快速滑模控制在滑动模态的设计综合了传统滑模控制与 Terminal 滑模控制的优点,同时在到达阶段也运用快速到达的概念。全局快速滑模控制具有以下特点:

(1) 全局快速滑模控制保证了系统在有限时间内到达滑模面,使系统状态在有限时间内迅速收敛到平衡状态。系统状态收敛到平衡状态的时间可以通过选取参数进行调整。

(2) 全局快速滑模控制的控制律是连续的,不含切换项,从而能消除抖振现象。

(3) 全局快速滑模控制对系统不确定性和干扰具有很好的鲁棒性,通过选取足够小的 q/p,可使系统状态到达滑模面足够小的邻域内,沿滑模面收敛到平衡状态。

9.3.2 全局快速滑模控制器的设计

考虑高阶单输入单输出非线性系统:

$$\begin{cases}\dot{x}_i=x_{i+1},i=1,2,\cdots,n-1\\ \dot{x}_n=f(\boldsymbol{x})+g(\boldsymbol{x})u(t)\end{cases} \tag{9.51}$$

其中 $f(\boldsymbol{x}),g(\boldsymbol{x})$ 是 $\mathbf{R}^n$ 域中的光滑函数,$g(\boldsymbol{x})\neq 0,u\in\mathbf{R}^1$。

一种具有递归结构的快速滑动模态表示为[2]

$$\begin{aligned}s_1&=\dot{s}_0+\alpha_0 s_0+\beta_0 s_0^{q_0/p_0}\\ s_2&=\dot{s}_1+\alpha_1 s_1+\beta_1 s_1^{q_1/p_1}\\ &\vdots\\ s_{n-1}&=\dot{s}_{n-2}+\alpha_{n-2}s_{n-2}+\beta_{n-2}s_{n-2}^{q_{n-2}/p_{n-2}}\end{aligned} \tag{9.52}$$

其中 $\alpha_i,\beta_i>0$,且 $q_i,p_i(q_i<p_i)(i=0,1,\cdots,n-2)$ 为正奇数。

设计全局快速滑模控制律为

$$u(t)=-\frac{1}{g(\boldsymbol{x})}\times\left(f(\boldsymbol{x})+\sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)}+\sum_{k=0}^{n-2}\beta_k\frac{d^{n-k-1}}{dt^{n-k-1}}s_k^{q_k/p_k}+\phi s_{n-1}+\gamma s_{n-1}^{q/p}\right)\tag{9.53}$$

其中 $s_0=x_1$。

在控制律式(9.53)下，系统状态沿 $\dot{s}_{n-1}=-\phi s_{n-1}-\gamma s_{n-1}^{q/p}$ 到达滑模面 $s_{n-1}=0$ 的时间 $t_{s_{n-1}}$ 为

$$t_{s_{n-1}}=\frac{p}{\phi(p-q)}\ln\frac{\phi[x_1(0)]^{(p-q)/p}+\gamma}{\gamma}\tag{9.54}$$

其中 $\phi,\gamma>0$，p 和 $q(q<p)$ 为正奇数。

9.3.3　全局快速滑模到达时间及稳定性分析

1. 到达时间分析

由式(9.52)可得

$$\dot{s}_{n-1}=\ddot{s}_{n-2}+\alpha_{n-2}\dot{s}_{n-2}+\beta_{n-2}\frac{\mathrm{d}}{\mathrm{d}t}s_{n-2}^{q_{n-2}/p_{n-2}}\tag{9.55}$$

由于 $s_i=\dot{s}_{i-1}+\alpha_{i-1}s_{i-1}+\beta_{i-1}s_{i-1}^{q_{i-1}/p_{i-1}}$，$i=n-1,n-2,\cdots,1$，$s_i$ 的 l 阶导数为

$$s_i^{(l)}=s_{i-1}^{(l+1)}-\alpha_{i-1}s_{i-1}^{(l)}+\beta_{i-1}\frac{\mathrm{d}^l}{\mathrm{d}t^l}s_{i-1}^{q_{i-1}/p_{i-1}}\tag{9.56}$$

则

$$\ddot{s}_{n-2}=\dddot{s}_{n-3}+\alpha_{n-3}\ddot{s}_{n-3}+\beta_{n-3}\frac{\mathrm{d}^2}{\mathrm{d}t^2}s_{n-3}^{q_{n-3}/p_{n-3}}\tag{9.57}$$

将式(9.57)代入式(9.55)，得

$$\dot{s}_{n-1}=\dddot{s}_{n-3}+\alpha_{n-3}\ddot{s}_{n-3}+\beta_{n-3}\frac{\mathrm{d}^2}{\mathrm{d}t^2}s_{n-3}^{q_{n-3}/p_{n-3}}+\alpha_{n-2}\dot{s}_{n-2}+\beta_{n-2}\frac{\mathrm{d}}{\mathrm{d}t}s_{n-2}^{q_{n-2}/p_{n-2}}$$

通过递推，得

$$\begin{aligned}\dot{s}_{n-1}&=s_0^{(n)}+\sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)}+\sum_{k=0}^{n-2}\beta_k\frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}}s_k^{q_k/p_k}\\&=\dot{x}_n+\sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)}+\sum_{k=0}^{n-2}\beta_k\frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}}s_k^{q_k/p_k}\end{aligned}\tag{9.58}$$

将式(9.51)代入式(9.58)，得

$$\dot{s}_{n-1}=f(\boldsymbol{x})+g(\boldsymbol{x})u(t)+\sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)}+\sum_{k=0}^{n-2}\beta_k\frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}}s_k^{q_k/p_k}\tag{9.59}$$

将控制律式(9.53)代入式(9.59)，得

$$\dot{s}_{n-1}=-\phi s_{n-1}-\gamma s_{n-1}^{q/p}\tag{9.60}$$

解微分方程式(9.60)，得

$$t_{s_{n-1}} = \frac{p}{\phi(p-q)}\ln\frac{\phi[x_1(0)]^{(p-q)/p}+\gamma}{\gamma}$$

2. 稳定性分析

定义 Lyapunov 函数：

$$V = \frac{1}{2}s_{n-1}^2 \tag{9.61}$$

则由式(9.60)得

$$\dot{V} = s_{n-1}\dot{s}_{n-1} = -\phi s_{n-1}^2 - \gamma s_{n-1}^{(q+p)/p}$$

由于$(p+q)$为偶数，所以$\dot{V}\leqslant 0$，系统稳定。

9.3.4 仿真实例

考虑二阶单输入单输出非线性系统：

$$\begin{cases}\dot{x}_1 = x_2\\ \dot{x}_2 = \cos x_1 + (x_1^2+1)u\end{cases}$$

快速滑模面设计为

$$s_1 = \dot{s}_0 + \alpha_0 s_0 + \beta_0 s_0^{q_0/p_0} = 0$$

其中 $s_0 = x_1$。

根据式(9.53)，得全局快速滑动模态控制律为

$$u = \frac{1}{x_1^2+1}\left(-\cos x_1 - \alpha_0\dot{x}_1 - \beta_0\frac{q}{p}x_1^{(q_0-p_0)/p_0}\dot{x}_1 - \phi s_1 - \gamma s_1^{q/p}\right)$$

设系统的初始状态为[5 0]，控制律参数取 $\alpha_0=2$，$\beta_0=1$，$p_0=9$，$q_0=5$，$\phi=10$，$\gamma=10$，$p=3$，$q=1$。根据式(9.46)得收敛时间为 $t_s=1.8306$。仿真结果如图 9-17～图 9-19 所示。

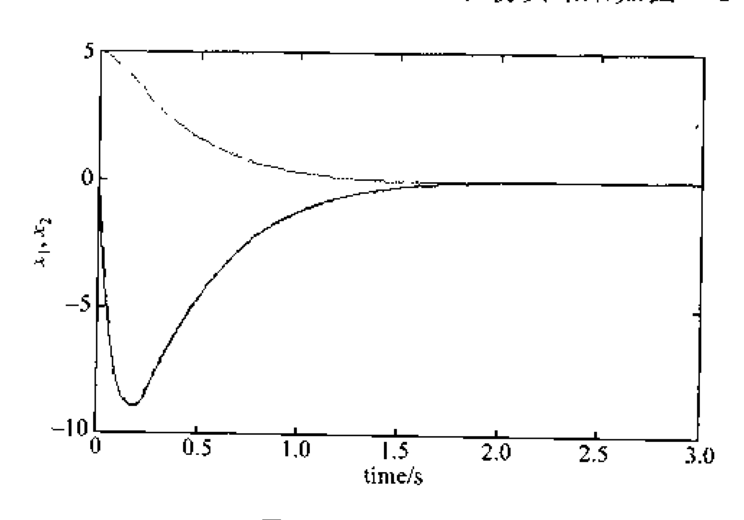


图 9-17 状态响应

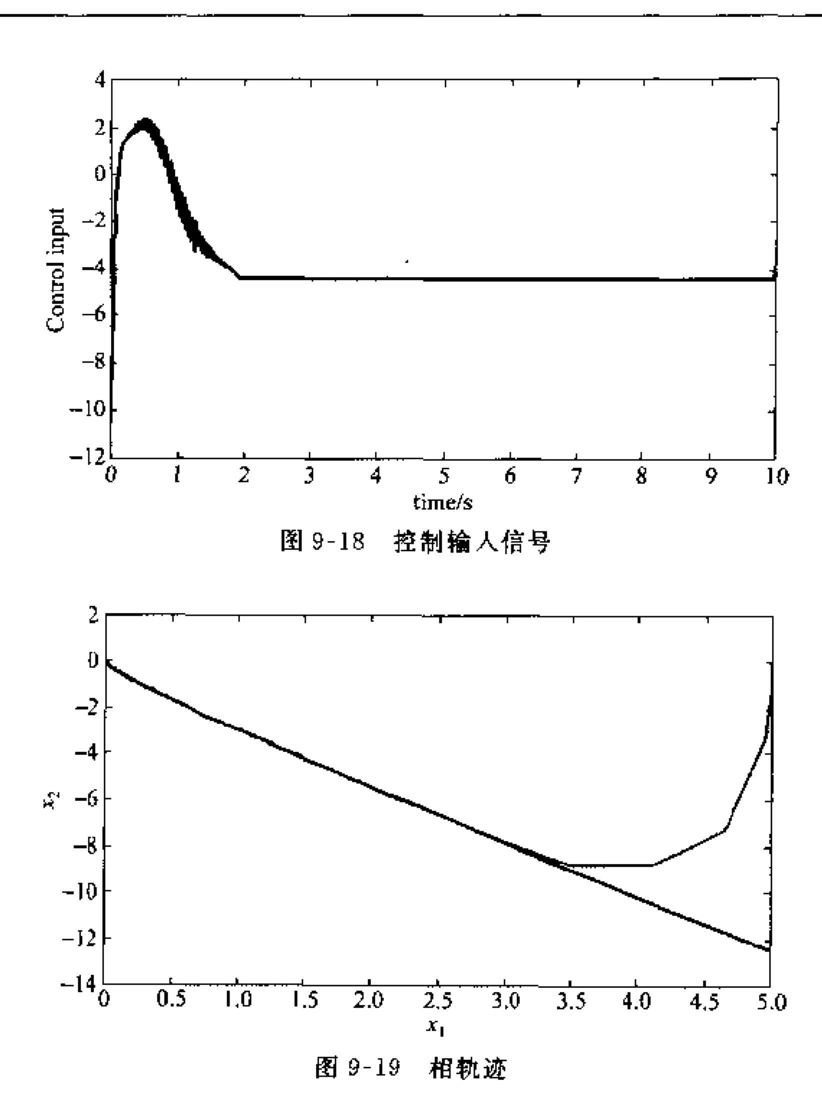


图 9-18　控制输入信号

图 9-19　相轨迹

仿真程序：

(1) 主程序(如图 9-20 所示)：chap9_4sim.mdl

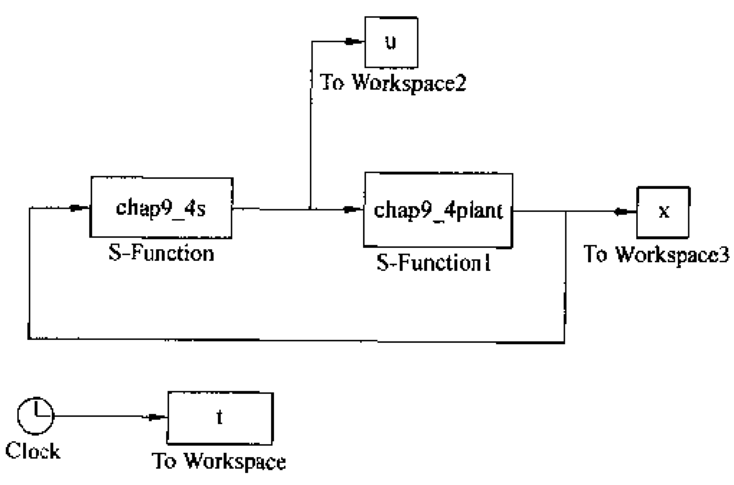


图 9-20　主程序图

(2) S 函数控制子程序：chap9_4s.m

```
% S-function for continuous state equation
```

```
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
 %Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
 %Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
 %Unhandled flags
  case {2, 4, 9 }
    sys = [];
 %Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

 %mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)
alfa0 = 2;
beta0 = 1;
p0 = 9;
q0 = 5;
fai = 10;
gama = 10;
p = 3;
q = 1;

x(1) = u(1);
x(2) = u(2);

s0 = x(1);
s1 = x(2) + alfa0 * s0 + beta0 * s0^(q0/p0);

%The time to reach the equilibrium x = 0
x0 = 5;
T = (alfa0 * x0^((p0 - q0)/p0) + beta0)/beta0;
ts = p0/(alfa0 * (p0 - q0)) * log(T);

z = sign(s1) * (abs(s1))^(q/p);
```

```
ut = 1/(x(1)^2 + 1) * ( - cos(x(1)) - alfa0 * x(2) - beta0 * q/p * x(1)^((q0 - p0)/p0) * x(2) - fai
 * s1 - gama * z);

sys(1) = ut;
```

(3) S 函数被控对象子程序：chap9_4plant.m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [5,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)

sys(1) = x(2);
sys(2) = cos(x(1)) + (x(1)^2 + 1) * u;

function sys = mdlOutputs(t,x,u)

sys(1) = x(1);
sys(2) = x(2);
```

(4) 作图子程序：chap9_4plot.m

```
close all;
```

```
figure(1);
plot(t,x(:,1),'r',t,x(:,2),'b');
xlabel('time(s)');ylabel('State response');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
alfa0 = 2;
beta0 = 1;
p0 = 9;
q0 = 5;
plot(x(:,1),x(:,2),'r',x(:,1), - alfa0 * x(:,1) - beta0 * abs(x(:,1)).^(q0/p0). * sign(x(:,
1)),'b');
xlabel('x1');ylabel('x2');
```

9.3.5 位置跟踪控制器的设计

设位置指令为 r，则

$$s_0 = r - \boldsymbol{x}(1)$$

$$s_0^{(n)} = r^{(n)} - \boldsymbol{x}(1)^{(n)} = r^{(n)} - \dot{x}_n$$

由式(9.58)可知

$$\begin{aligned}\dot{s}_{n-1} &= s_0^{(n)} + \sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)} + \sum_{k=0}^{n-2}\beta_k \frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}} s_k^{q_k/p_k} \\ &= r^{(n)} - \dot{x}_n + \sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)} + \sum_{k=0}^{n-2}\beta_k \frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}} s_k^{q_k/p_k} \\ &= r^{(n)} - f(\boldsymbol{x}) - g(\boldsymbol{x})u + \sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)} + \sum_{k=0}^{n-2}\beta_k \frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}} s_k^{q_k/p_k}\end{aligned}$$

由式(9.60)可知

$$\dot{s}_{n-1} = -\phi s_{n-1} - \gamma s_{n-1}^{q/p}$$

由上两式联立得位置控制律为

$$u(t) = \frac{1}{g(\boldsymbol{x})}\left[r^{(n)} - f(\boldsymbol{x}) + \sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)} + \sum_{k=0}^{n-2}\beta_k \frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}} s_k^{q_k/p_k} + \phi s_{n-1} + \gamma s_{n-1}^{q/p}\right] \tag{9.62}$$

9.3.6 仿真实例

针对如下二阶系统：

$$\begin{cases}\dot{x}_1 = x_2 \\ \dot{x}_2 = -25x_2 + 133u\end{cases}$$

则 $n=2$，$s_0 = r - \boldsymbol{x}(1)$，$s_1 = \dot{s}_0 + \alpha_0 s_0 + \beta_0 s_0^{\frac{q_0}{p_0}}$，由式(9.62)得控制器为

$$u(t) = \frac{1}{g(\boldsymbol{x})}\left[\ddot{r} - f(\boldsymbol{x}) + \alpha_0\dot{s}_0 + \beta_0\frac{\mathrm{d}}{\mathrm{d}t}s_0^{\frac{q_0}{p_0}} + \phi s_1 + \gamma s_1^{q/p}\right]$$

指令信号取 $r=\sin(6\pi t)$，控制律参数取 $\alpha_0=2$，$\beta_0=1$，$p_0=9$，$q_0=5$，$\phi=100$，$\gamma=10$，$p=3$，$q=1$。仿真结果如图 9-21 和图 9-22 所示。

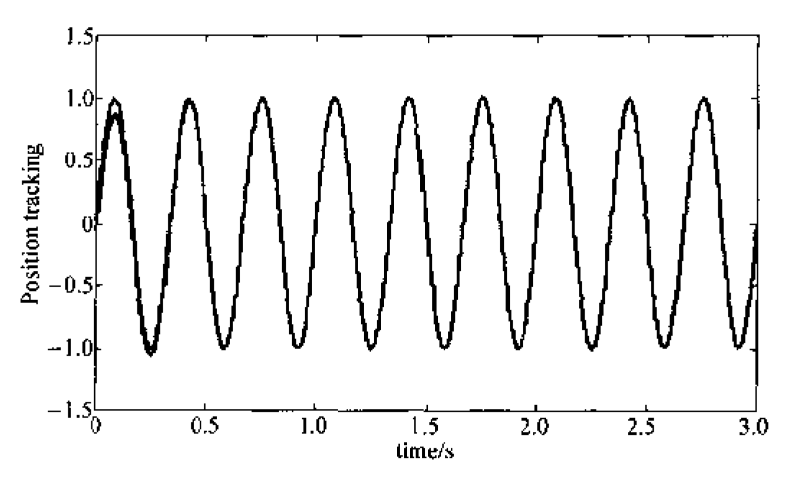


图 9-21 位置跟踪

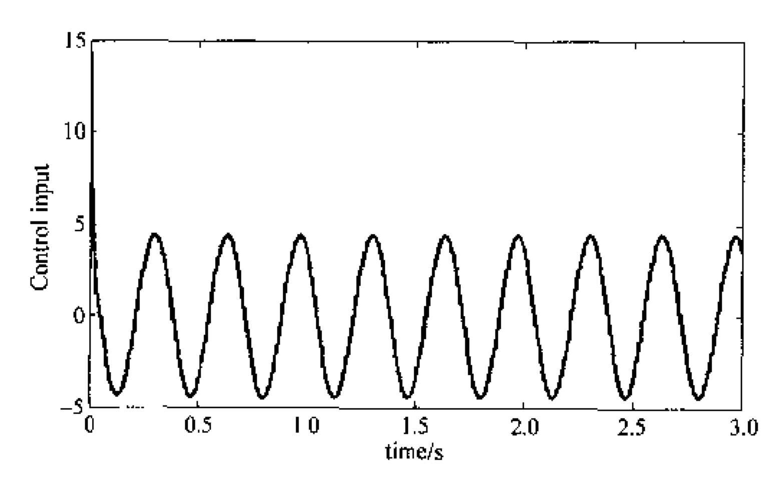


图 9-22 控制输入信号

仿真程序：

(1) Simulink 主程序(如图 9-23 所示)：chap9_5sim. mdl

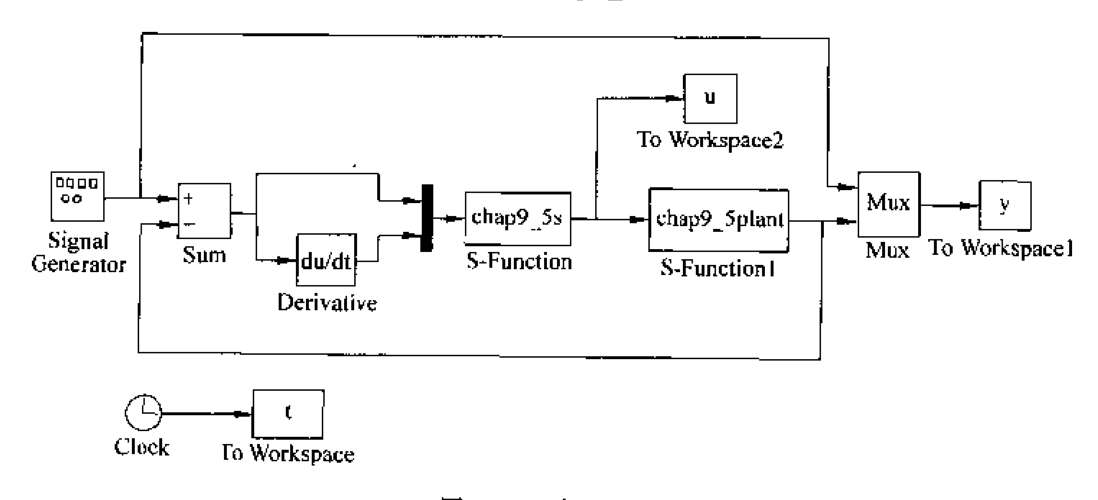


图 9-23 主程序图

(2) S 函数控制子程序:chap9_5s.m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 0;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 1;
sizes.NumInputs       = 2;
sizes.DirFeedthrough  = 1;
sizes.NumSampleTimes  = 0;

sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)

r = sin(3 * 2 * pi * t);
dr = 3 * 2 * pi * cos(3 * 2 * pi * t);
ddr = - (3 * 2 * pi)^2 * sin(3 * 2 * pi * t);

e = u(1);
de = u(2);
x2 = dr - de;

fx = - 25 * x2;
gx = 133;

alfa0 = 2;
beta0 = 1;
p0 = 9;q0 = 5;
```

```
p = 3;q = 1;

fai = 100;
gama = 10;

s0 = e;
ds0 = de;
z0 = abs(s0)^(q0/p0) * sign(s0);
s1 = ds0 + alfa0 * s0 + beta0 * z0;
z1 = abs(s1)^(q/p) * sign(s1);

ut = 1/gx * (ddr - fx + alfa0 * ds0 + beta0 * q0/p0 * z0 * ds0 + fai * s1 + gama * z1);

sys(1) = ut;
```

(3) S 函数被控对象子程序:chap9_5plant. m

```
%S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys  = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes  = simsizes;
sizes.NumContStates   = 2;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 1;
sizes.NumInputs       = 1;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 0;

sys = simsizes(sizes);
x0 = [0,0];
str = [];
ts = [];
```

```
function sys = mdlDerivatives(t,x,u)

sys(1) = x(2);
sys(2) = - 25 * x(2) + 133 * u;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
```

(4) 作图子程序:chap9_5plot. m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');
```

9.3.7 具有鲁棒性的全局快速滑模控制

考虑不确定系统:

$$\begin{cases}\dot{x}_i = x_{i+1} \\ \dot{x}_n = f_0(\boldsymbol{x}) + g(\boldsymbol{x})u + d(\boldsymbol{x})\end{cases} \tag{9.63}$$

其中 $i=1,2,\cdots,n-1$, $f_0(\boldsymbol{x})$, $g(\boldsymbol{x})$是 $\mathbf{R}^n$ 域中的光滑函数,$g(\boldsymbol{x})\neq 0$, $u\in\mathbf{R}^1$。$d(\boldsymbol{x})$表示系统的参数不确定性和外部干扰的总和,设$|d(\boldsymbol{x})|\leqslant L$。

鲁棒控制律设计为

$$u(t) = -\frac{1}{g(\boldsymbol{x})}\left[f_0(\boldsymbol{x}) + \sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)} + \sum_{k=0}^{n-2}\beta_k \frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}} s_k^{q_k/p_k} + \phi s_{n-1} + \gamma s_{n-1}^{q/p}\right] \tag{9.64}$$

其中 $s_0 = x_1$, $\phi,\gamma>0$, p 和 q 为正奇数($q<p$)。

在控制律式(9.64)作用下,系统状态沿 $\dot{s}_{n-1} = -\phi s_{n-1} - \gamma' s_{n-1}^{q/p}$ 在有限时间 t'_{n-1} 内到达滑模面 $s_{n-1}=0$ 的 Δ 邻域内。则有

$$t'_{n-1} \leqslant \frac{p}{\phi(p-q)}\ln\frac{\phi x_1(0)^{(p-q)/p}+\eta}{\eta} \tag{9.65}$$

其中

$$\begin{aligned}\gamma' &= \gamma - \frac{d(\boldsymbol{x})}{s_{n-1}^{q/p}} \\ \gamma &= \frac{L}{|s_{n-1}^{q/p}|} + \eta \quad \eta > 0 \\ \Delta &= \left\{\boldsymbol{x} : |s_{n-1}| \leqslant \left(\frac{L}{\gamma}\right)^{p/q}\right\}\end{aligned} \tag{9.66}$$

1. 稳定性分析

定义 Lyapunov 函数

$$V = \frac{1}{2}s_{n-1}^2$$

由式(9.59)并结合式(9.63)可知

$$\dot{s}_{n-1} = f_0(\boldsymbol{x}) + g(\boldsymbol{x})u(t) + d(\boldsymbol{x}) + \sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)} + \sum_{k=0}^{n-2}\beta_k \frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}} s_k^{q_k/p_k}$$

将控制律式(9.64)代入上式得

$$\begin{aligned}\dot{s}_{n-1} &= f_0(\boldsymbol{x}) - f_0(\boldsymbol{x}) - \phi s_{n-1} - \gamma s_{n-1}^{q/p} + d(\boldsymbol{x}) \\ &= -\phi s_{n-1} - \left(\gamma - \frac{d(\boldsymbol{x})}{s_{n-1}^{q/p}}\right)s_{n-1}^{q/p} = -\phi s_{n-1} - \gamma' s_{n-1}^{q/p}\end{aligned} \tag{9.67}$$

其中 $\gamma' = \gamma - \frac{d(\boldsymbol{x})}{s_{n-1}^{q/p}}$。

由 $\gamma > \frac{L}{|s_{n-1}^{q/p}|}$ 得

$$\gamma' > \frac{L}{|s_{n-1}^{q/p}|} - \frac{d(\boldsymbol{x})}{s_{n-1}^{q/p}} \geqslant \frac{L}{|s_{n-1}^{q/p}|} - \frac{|d(\boldsymbol{x})|}{|s_{n-1}^{q/p}|} \geqslant 0$$

则

$$\dot{V} = s_{n-1}\dot{s}_{n-1} = -\phi s_{n-1}^2 - \gamma' s_{n-1}^{(q+p)/p} \tag{9.68}$$

由于$(p+q)$为偶数，所以 $\dot{V} \leqslant 0$，系统稳定。

2. 滑模到达时间分析

由式(9.67)得

$$\dot{s}_{n-1} = -\phi s_{n-1} - \gamma' s_{n-1}^{q/p} \tag{9.69}$$

解微分方程(9.69)，得

$$t_{s_{n-1}} = \frac{p}{\phi(p-q)}\ln\frac{\phi[x_1(0)]^{(p-q)/p} + \gamma'}{\gamma'}$$

由于 $\gamma' \geqslant \eta$，则

$$\ln\frac{\phi[x_1(0)]^{(p-q)/p} + \gamma'}{\gamma'} \leqslant \ln\frac{\phi[x_1(0)]^{(p-q)/p} + \eta}{\eta}$$

故到达时间为

$$t_{s_{n-1}} \leqslant \frac{p}{\phi(p-q)}\ln\frac{\phi[x_1(0)]^{(p-q)/p} + \eta}{\eta}$$

又由 $\gamma > \frac{L}{|s_{n-1}^{q/p}|}$ 得

$$|s_{n-1}| > \left(\frac{L}{\gamma}\right)^{p/q} \tag{9.70}$$

式(9.70)为 Δ 的约束条件。由式(9.70)即可求出系统状态到达滑模面 $s_{n-1}=0$ 的邻域 Δ。

由于 Δ 邻域受 $|s_{n-1}| < \left(\frac{L}{\gamma}\right)^{p/q}$ 约束，只要选取足够大的 γ 和 p/q，就可使得滑模面 $s_{n-1}=0$ 的邻域 Δ 足够小。例如，取 $L=1, q/p=1/9, \gamma=2$，则 $\Delta = |s_{n-1}| \leqslant \left(\frac{L}{\gamma}\right)^{p/q} = 2\times 10^{-3}$。所以系统的性能主要依赖于 L, p, q 值的选择。

3. 位置滑模控制器的设计

设位置指令为 r，对应于控制律式(9.64)，位置控制律设计为

$$u(t)=\frac{1}{g(x)}\left[r^{(n)}-f_0(x)+\sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)}+\sum_{k=0}^{n-2}\beta_k\frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}}s_k^{q_k/p_k}+\phi s_{n-1}+\gamma s_{n-1}^{q/p}\right]\quad(9.71)$$

稳定性分析：

定义 Lyapunov 函数

$$V=\frac{1}{2}s_{n-1}^2$$

$$s_0=r-x(1),s_0^{(n)}=r^{(n)}-x(1)^{(n)}=r^{(n)}-\dot{x}_n$$

由式(9.58)并结合式(9.63)得

$$\begin{aligned}\dot{s}_{n-1}&=s_0^{(n)}+\sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)}+\sum_{k=0}^{n-2}\beta_k\frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}}s_k^{q_k/p_k}\\&=r^{(n)}-\dot{x}_n+\sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)}+\sum_{k=0}^{n-2}\beta_k\frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}}s_k^{q_k/p_k}\\&=r^{(n)}-f_0(x)-g(x)u-d(x)+\sum_{k=0}^{n-2}\alpha_k s_k^{(n-k-1)}-\sum_{k=0}^{n-2}\beta_k\frac{\mathrm{d}^{n-k-1}}{\mathrm{d}t^{n-k-1}}s_k^{q_k/p_k}\end{aligned}$$

将控制律式(9.71)代入上式，得

$$\dot{s}_{n-1}=-\phi s_{n-1}-\gamma s_{n-1}^{q/p}-d(x)=-\phi s_{n-1}-L\mathrm{sgn}(s_{n-1})-\eta s_{n-1}^{q/p}-d(x)$$

$$\dot{V}=s_{n-1}\dot{s}_{n-1}=-\phi s_{n-1}^2-\eta s_{n-1}^{(p+q)/p}-d(x)s_{n-1}-L\mid s_{n-1}\mid\leqslant-\phi s_{n-1}^2-\eta s_{n-1}^{(p+q)/p}\leqslant 0$$

则系统稳定。

9.3.8 仿真实例

考虑二阶系统：

$$\begin{cases}\dot{x}_1=x_2\\\dot{x}_2=-25x_2+133u+\mathrm{d}t\end{cases}$$

其中 $\mathrm{d}t=1.5\sin(t)$。

位置指令信号取 $r=\sin(6\pi t)$，则 $s_0=r-x(1)$，$s_1=s_0+\alpha_0 s_0+\beta_0 s_0^{q_0/p_0}$。

根据式(9.71)，位置控制器设计为

$$u(t)=\frac{1}{g(x)}\left[\ddot{r}-f_0(x)+\alpha_0\dot{s}_0+\beta_0\frac{\mathrm{d}}{\mathrm{d}t}s_0^{q_0/p_0}+\phi s_1+\gamma s_1^{q/p}\right]$$

其中 γ 按式(9.66)选取，$L=2.0$。

控制律参数取 $\alpha_0=2,\beta_0=1,p_0=9,q_0=5,\phi=100,p=3,q=1,\gamma=\frac{L}{|s_1^{q/p}|}+\eta,\eta=1.0$。仿真结果如图 9-24～图 9-26 所示。

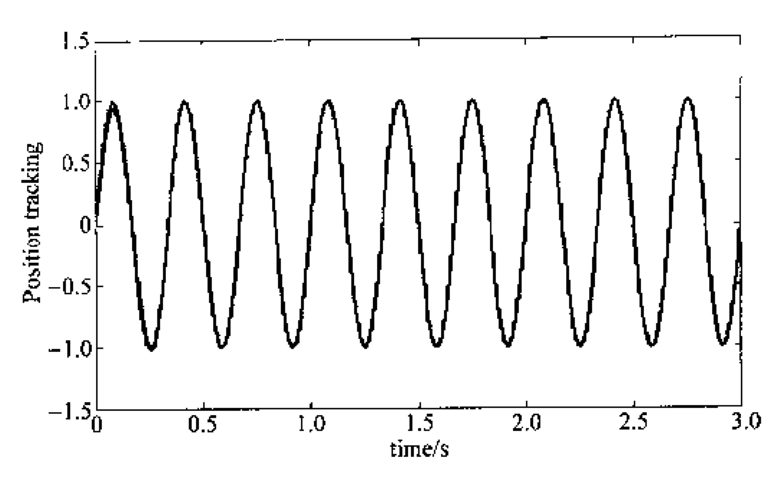


图 9-24　位置跟踪结果

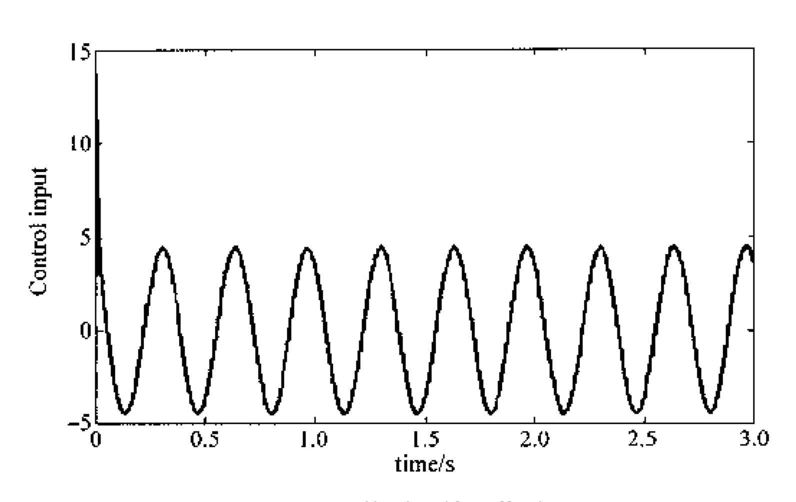


图 9-25　控制器输入信号

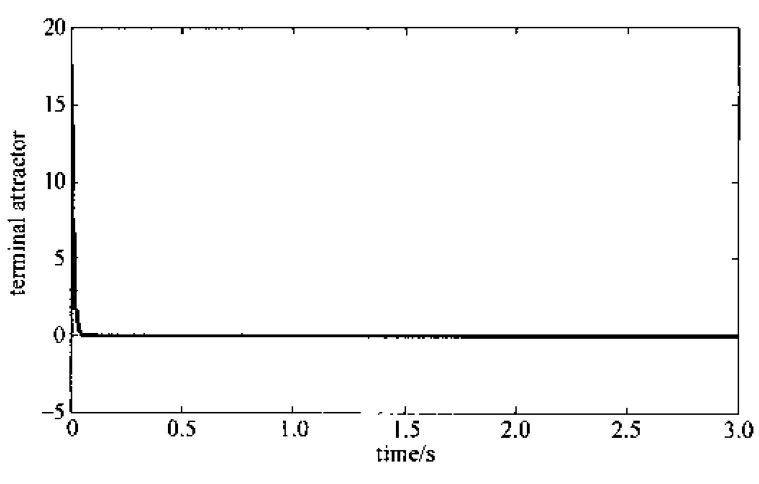


图 9-26　滑模函数的变化

仿真程序：

(1) Simulink 主程序(如图 9-27 所示)：chap9_6sim. mdl

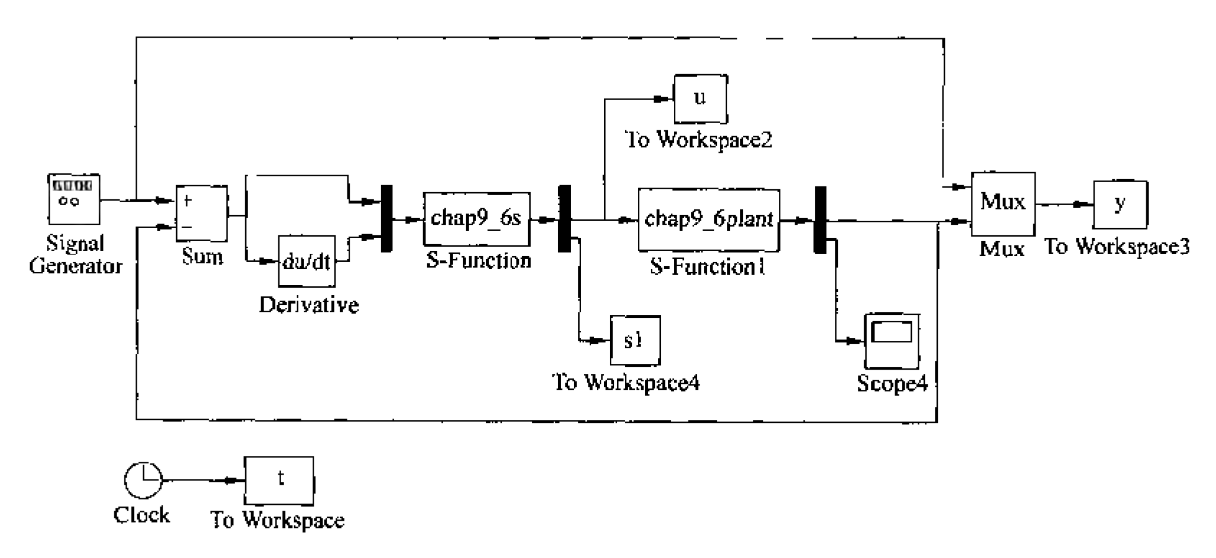


图 9-27 主程序图

(2) S 函数控制子程序：chap9_6s.m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];
```

```
function sys = mdlOutputs(t,x,u)
L = 2;

r = sin(3 * 2 * pi * t);
dr = 3 * 2 * pi * cos(3 * 2 * pi * t);
ddr = -(3 * 2 * pi)^2 * sin(3 * 2 * pi * t);

e = u(1);
de = u(2);
x2 = dr - de;

fx = -25 * x2;
gx = 133;

alfa0 = 2;
beta0 = 1;
p0 = 9;q0 = 5;
p = 3;q = 1;
fai = 100;

s0 = e;
ds0 = de;
z0 = abs(s0)^(q0/p0) * sign(s0);
s1 = ds0 + alfa0 * s0 + beta0 * z0;
z1 = abs(s1)^(q/p) * sign(s1);

Ts = (alfa0 * 0.1^((p0 - q0)/p0) + beta0)/beta0;
ts = p0/(alfa0 * (p0 - q0)) * log(Ts);

xite = 1.0;
T = fai * 0.1^((p - q)/p) + xite;
t_s1 = p/(fai * (p - q)) * log(T)/xite;    % Time of s1 = 0

gama = L/abs(z1) + xite;

ut = 1/gx * (ddr - fx + alfa0 * ds0 + beta0 * q0/p0 * z0 * ds0 + fai * s1 + gama * z1);

sys(1) = ut;
sys(2) = s1;
```

(3) S 函数被控对象子程序:chap9_6plant.m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
```

```
      sys = mdlDerivatives(t,x,u);
  % Outputs
    case 3,
      sys = mdlOutputs(t,x,u);
  % Unhandled flags
    case {2, 4, 9 }
      sys = [ ];
  % Unexpected flags
    otherwise
      error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 2;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 1;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 0;

sys = simsizes(sizes);
x0 = [0.1,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
dt = 1.5 * sin(t);
L = 2;    % L = max(abs(dt))

sys(1) = x(2);
sys(2) = - 25 * x(2) + 133 * u + dt;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(4) 作图子程序:chap9_6plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,s1(:,1),'r');
```

```
xlabel('time(s)');ylabel('terminal attractor');
```

9.4 非奇异 Terminal 滑模控制

考虑如下二阶不确定非线性动态系统：

$$\begin{cases}\dot{x}_1 = x_2 \\ \dot{x}_2 = f(\boldsymbol{x}) + g(\boldsymbol{x}) + b(\boldsymbol{x})u\end{cases} \tag{9.72}$$

其中 $\boldsymbol{x}=[x_1\ x_2]^{\mathrm{T}}$，$b(x)\neq 0$，$g(\boldsymbol{x})$代表不确定性及外部干扰，$g(\boldsymbol{x})\leqslant l_g$。

9.4.1 普通 Terminal 滑模控制

1. 控制器设计

滑模函数设计为[3]

$$s = x_2 + \beta x_1^{q/p} \tag{9.73}$$

其中 $\beta>0$，p 和 q（$p>q$）为正奇数。

控制器设计为

$$u = -b^{-1}(\boldsymbol{x})\left(f(\boldsymbol{x}) + \beta\frac{q}{p}x_1^{\frac{q}{p}-1}x_2 + (l_g+\eta)\operatorname{sgn}(s)\right) \tag{9.74}$$

其中 $\eta>0$。

稳定性分析：

$$\begin{aligned}\dot{s} &= \dot{x}_2 + \beta\frac{q}{p}x_1^{\frac{q}{p}-1}\dot{x}_1 = f(\boldsymbol{x}) + g(\boldsymbol{x}) + b(\boldsymbol{x})u + \beta\frac{q}{p}x_1^{\frac{q}{p}-1}\dot{x}_1 \\ &= f(\boldsymbol{x}) + g(\boldsymbol{x}) - f(\boldsymbol{x}) - \beta\frac{q}{p}x_1^{\frac{q}{p}-1}x_2 - (l_g+\eta)\operatorname{sgn}(s) + \beta\frac{q}{p}x_1^{\frac{q}{p}-1}\dot{x}_1 \\ &= g(\boldsymbol{x}) - (l_g+\eta)\operatorname{sgn}(s)\end{aligned}$$

$$s\dot{s} = sg(\boldsymbol{x}) - (l_g+\eta)|s| \leqslant -\eta|s|$$

由式(9.74)可见，$\frac{q}{p}-1<0$，当 $x_1=0$，$x_2\neq 0$ 时，普通 Terminal 滑模控制器存在奇异问题。

2. 有限到达间分析

设 $s(0)\neq 0$ 到 $s=0$ 的时间为 t_r。当 $t=t_r$ 时，$s(t_r)=0$，$s\cdot\dot{s}=-\eta|s|$。

由 $s\dot{s}\leqslant -\eta|s|$ 得：

当 $s\geqslant 0$ 时，

$$\dot{s}\leqslant -\eta$$

$$\int_{s(0)}^{s(t_r)}\mathrm{d}s \leqslant \int_0^{t_r} -\eta\mathrm{d}t$$

$$s(t_r) - s(0) \leqslant -\eta t_r$$

$$t_r \leqslant \frac{s(0)}{\eta}$$

同理，$s\leqslant 0$ 时，$t_r\leqslant -\frac{s(0)}{\eta}$

即
$$t_r \leqslant \frac{|s(0)|}{\eta} \tag{9.75}$$

设 $x_1(t_r)\neq 0$ 到 $x_1(t_s+t_r)=0$ 的时间为 t_s，在此阶段，$s=0$，即

$$x_2+\beta x_1^{q/p}=0$$

$$\dot{x}_1=-\beta x_1^{\frac{q}{p}}$$

对上式进行积分，得：

$$\int_{x_1(t_r)}^{0} x_1^{\frac{q}{p}}\mathrm{d}x_1=\int_{t_r}^{t_r+t_s}-\beta \mathrm{d}t$$

$$-\frac{p}{p-q}x_1^{1-\frac{q}{p}}(t_r)=-\beta t_s$$

$$t_s=\frac{p}{\beta(p-q)}|x_1(t_r)|^{1-\frac{q}{p}} \tag{9.76}$$

9.4.2 非奇异 Terminal 滑模控制器的设计

为了解决普通 Terminal 滑模控制的奇异问题，提出了非奇异 Terminal 滑模控制方法[1]。非奇异滑模面为

$$s=x_1+\frac{1}{\beta}x_2^{p/q} \tag{9.77}$$

其中 $\beta>0$，p 和 $q(p>q)$ 为正奇数。

非奇异滑模控制器设计为

$$u=-b^{-1}(\boldsymbol{x})\left(f(\boldsymbol{x})+\beta\frac{q}{p}x_2^{2-p/q}+(l_g+\eta)\operatorname{sgn}(s)\right) \tag{9.78}$$

其中 $1<p/q<2$，$\eta>0$。

稳定性分析：

$$\dot{s}=\dot{x}_1+\frac{1}{\beta}\frac{p}{q}x_2^{\frac{p}{q}-1}\dot{x}_2=x_2+\frac{1}{\beta}\frac{p}{q}x_2^{\frac{p}{q}-1}[f(\boldsymbol{x})+g(\boldsymbol{x})+b(\boldsymbol{x})u]$$

$$=x_2+\frac{1}{\beta}\frac{p}{q}x_2^{\frac{p}{q}-1}\left[f(\boldsymbol{x})+g(\boldsymbol{x})-f(\boldsymbol{x})-\beta\frac{q}{p}x_2^{2-p/q}-(l_g+\eta)\operatorname{sgn}(s)\right]$$

$$=\frac{1}{\beta}\frac{p}{q}x_2^{\frac{p}{q}-1}[g(\boldsymbol{x})-(l_g+\eta)\operatorname{sgn}(s)]$$

$$s\dot{s}=\frac{1}{\beta}\frac{p}{q}x_2^{\frac{p}{q}-1}[sg(\boldsymbol{x})-(l_g+\eta)|s|]$$

由于 $1<\frac{p}{q}<2$，则 $0<\frac{p}{q}-1<1$，又由于 $\beta>0$，p 和 $q(p>q)$ 为正奇数，则

$$x_2^{\frac{p}{q}-1}>0(x_2\neq 0\text{ 时})$$

$$s\dot{s}\leqslant\frac{1}{\beta}\frac{p}{q}x_2^{\frac{p}{q}-1}(-\eta|s|)=-\frac{1}{\beta}\frac{p}{q}x_2^{\frac{p}{q}-1}\eta|s|=-\eta'|s|$$

其中 $\eta'=\frac{1}{\beta}\frac{p}{q}x_2^{\frac{p}{q}-1}\eta>0(x_2\neq 0$ 时)。

可见，当 $x_2\neq 0$ 时，控制器满足 Lyapunov 稳定条件。

将控制器式(9.78)代入式(9.72)，得

$$\dot{x}_2 = -\beta\frac{q}{p}x_2^{2-p/q} + g(\boldsymbol{x}) - (l_g + \eta)\mathrm{sgn}(s)$$

当 $x_2=0$ 时，有

$$\dot{x}_2 = g(\boldsymbol{x}) - (l_g + \eta)\mathrm{sgn}(s)$$

当 $s>0$ 时，$\dot{x}_2 \leqslant -\eta$，当 $s<0$ 时，$\dot{x}_2 \geqslant -\eta$，系统的相轨迹如图 9-28 所示。由相轨迹可见，当 $x_2=0$ 时，在有限时间内实现 $s=0$。

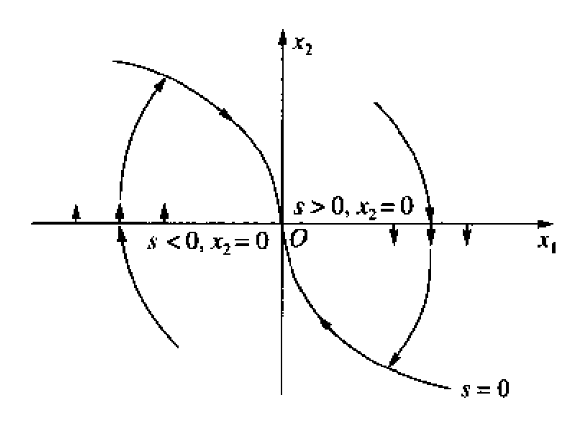


图 9-28　系统的相轨迹

9.4.3　仿真实例

考虑二阶单输入单输出非线性系统：

$$\begin{cases}\dot{x}_1 = x_2\\ \dot{x}_2 = 0.1\sin(20t) + u\end{cases}$$

取 $q=3$，$p=5$，将 NTSM 和 TSM 的滑模面分别设计为

$$s_{\mathrm{TSM}} = x_2 + x_1^{3/5}$$

$$s_{\mathrm{NTSM}} = x_1 + x_2^{5/3}$$

设系统的初始状态为[0.1 0]，控制律参数取 lg=0.015，$\beta=1.0$，$\eta=0.20$。取 $M=1$ 和 $M=2$，分别对应控制律(9.74)和(9.78)，仿真结果如图 9-29～图 9-34 所示。如果在控制

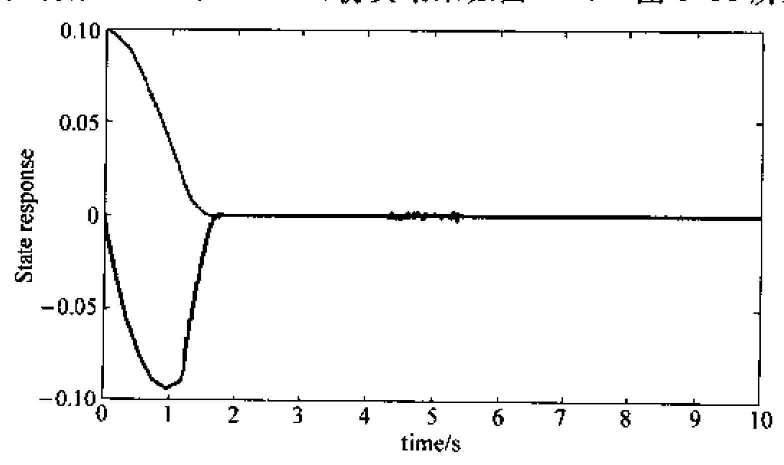


图 9-29　状态响应($M=1$；TSM)

律中对符号函数 sgn(s)采用饱和方法,可有效地消除抖振。

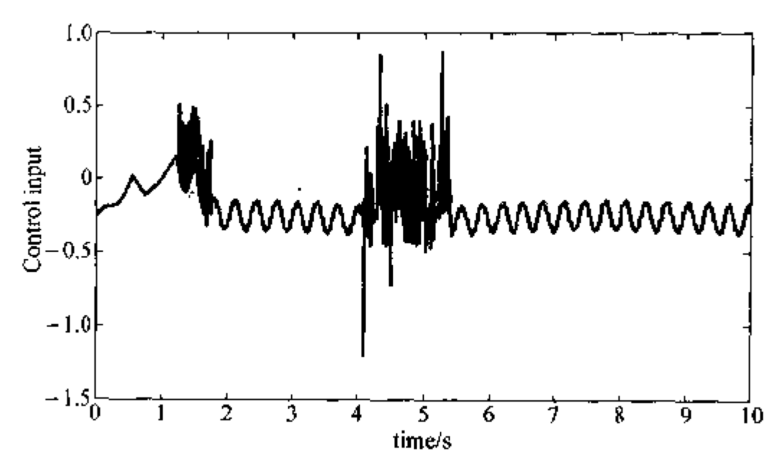


图 9-30 控制输入信号($M=1$:TSM)

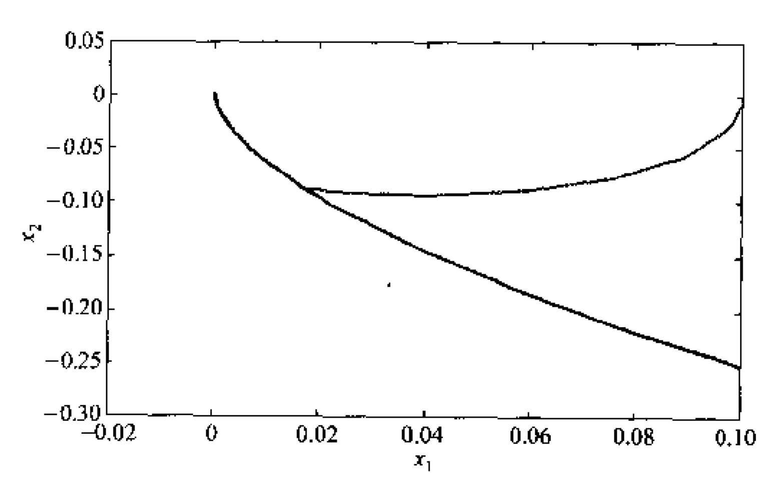


图 9-31 相轨迹($M=1$:TSM)

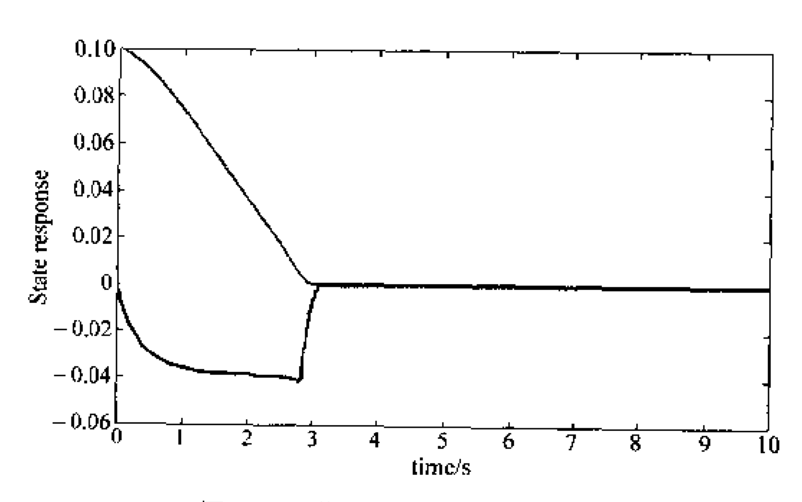


图 9-32 状态响应($M=2$:NTSM)

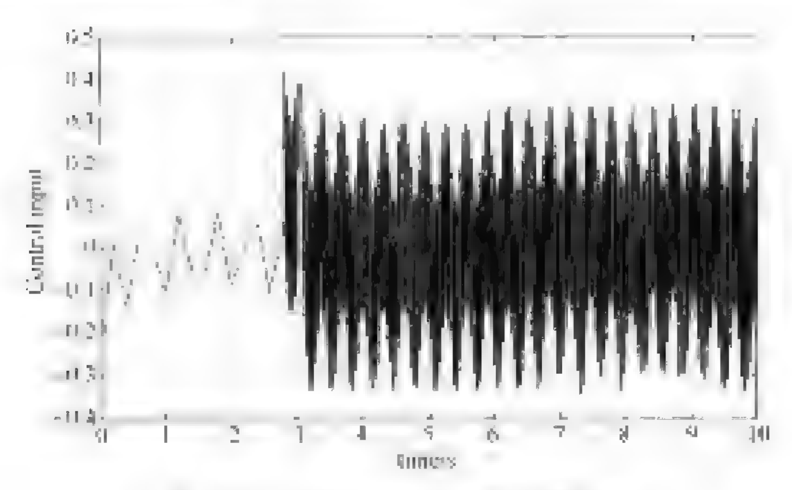


图 9-33　控制输入信号(M=2,NTSM)

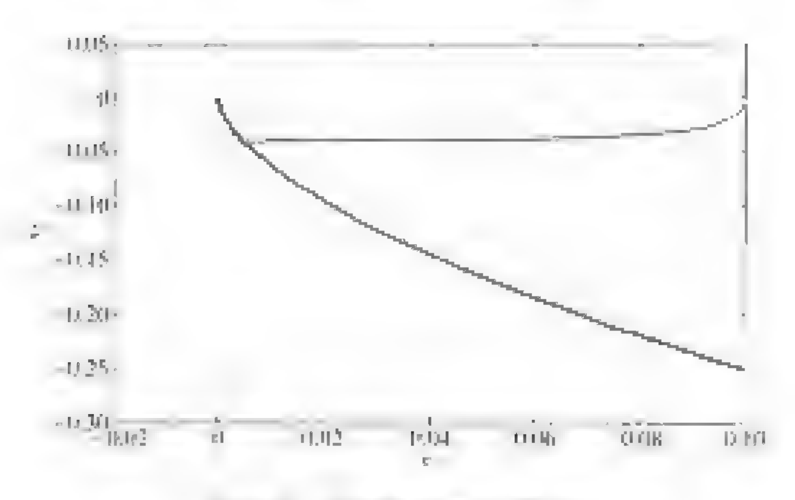


图 9-34　相轨迹(M=2,NTSM)

仿真程序:

(1) 主程序(如图 9-35 所示):chap9_7sim.mdl

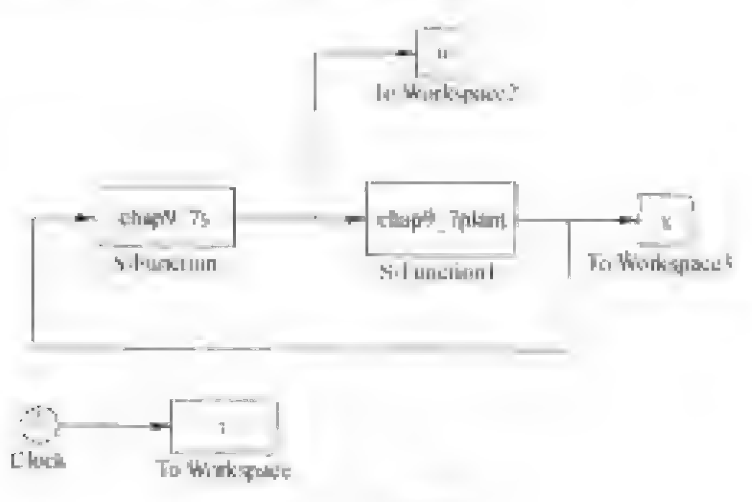


图 9-35　主程序图

(2) S 函数控制子程序:chap9_7s.m

```
%S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];
function sys = mdlOutputs(t,x,u)
x = u;

bx = 1.0;
fx = 0.1 * sin(20 * t);
gx = 0.12 * sin(t);

lg = 0.015;
beta = 1.0;
xite = 0.20;

q = 3;p = 5;

M = 2;
if M == 1          %TSM
   T1 = abs(x(1))^(q/p) * sign(x(1));
   T2 = abs(x(1))^(q/p - 1) * sign(x(1));
```

```
    s = x(2) + beta * T1;
    ut = - inv(bx) * (fx + beta * q/p * T2 * x(2) + (lg + xite) * sign(s));
elseif M == 2     % NTSM
    T1 = abs(x(2))^(p/q) * sign(x(2));
    T2 = abs(x(2))^(2 - p/q) * sign(x(2));
    s = x(1) + 1/beta * T1;
   ut = - inv(bx) * (fx + beta * q/p * T2 + (lg + xite) * sign(s));
end

sys(1) = ut;
```

(3) S 函数被控对象子程序：chap9_7plant. m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 2;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 1;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 0;

sys = simsizes(sizes);
x0 = [0,1,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)

sys(1) = x(2);
```

```
sys(2) = 0.1 * sin(20 * t) + u + 0.012 * sin(t);
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(4) 作图子程序:chap9_7plot.m

```
close all;

figure(1);
plot(t,x(:,1),'r',t,x(:,2),'b');
xlabel('time(s)');ylabel('State response');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
M = 2;
q = 3;p = 5;
if M == 1        % TSM
plot(x(:,1),x(:,2),'r',x(:,1), - (abs(x(:,1))).^(q/p). * sign(x(:,1)),'b');
elseif M == 2   % NTSM
plot(x(:,1),x(:,2),'r',x(:,1),(abs( - x(:,1))).^(q/p). * sign( - x(:,1)),'b');
end
xlabel('x1');ylabel('x2');
```

9.4.4 刚性机器人非奇异 Terminal 滑模控制

刚性机器人的动态方程为

$$M(q)\ddot{q} + C(q,\dot{q}) + g(q) = \tau(t) + d(t) \tag{9.79}$$

其中

$$M(q) = M_0(q) + \Delta M(q)$$
$$C(q,\dot{q}) = C_0(q,\dot{q}) + \Delta C(q,\dot{q})$$
$$g(q) = g_0(q) + \Delta g(q)$$

其中 $M_0(q)$,$C_0(q,\dot{q})$和 $g_0(q)$为机器人动态方程的估计项,$\Delta M(q)$,$\Delta C(q,\dot{q})$和 $\Delta g(q)$为机器人动态方程的不确定项。

方程(9.79)又可以写为

$$M_0(q)\ddot{q} + C_0(q,\dot{q}) + g_0(q) = \tau(t) + \rho(t) \tag{9.80}$$

其中

$$\rho(t) = - \Delta M(q)\ddot{q} - \Delta C(q,\dot{q}) - \Delta g(q) + d(t) \tag{9.81}$$

假设 $\| \rho(t) \| < b_0 + b_1 \| q \| + b_2 \| \dot{q} \|^2$。

设位置指令为 q_r,定义 $\varepsilon(t) = q - q_r$,则 $\dot{\varepsilon}(t) = \dot{q} - \dot{q}_r$。定义 $e(t) = [\varepsilon^{\mathrm{T}}(t)\ \dot{\varepsilon}^{\mathrm{T}}(t)]^{\mathrm{T}}$。

非奇异滑模面设计为

$$s = \varepsilon + C_1 \dot{\varepsilon}^{p/q} \tag{9.82}$$

其中 $C_1=\mathrm{diag}[c_{11}\ c_{12}\cdots c_{1n}]$。

控制律设计为

$$\boldsymbol{\tau}=\boldsymbol{\tau}_0+\boldsymbol{u}_0+\boldsymbol{u}_1 \tag{9.83}$$

其中

$$\boldsymbol{\tau}_0=\boldsymbol{C}_0(\boldsymbol{q},\dot{\boldsymbol{q}})+\boldsymbol{g}_0(\boldsymbol{q})+\boldsymbol{M}_0(\boldsymbol{q})\ddot{\boldsymbol{q}}_r \tag{9.84}$$

$$\boldsymbol{u}_0=-\frac{q}{p}\boldsymbol{M}_0(\boldsymbol{q})\boldsymbol{C}_1^{-1}\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{2-p/q}) \tag{9.85}$$

$$\begin{aligned}\boldsymbol{u}_1=&-\frac{[\boldsymbol{s}^{\mathrm{T}}\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})]^{\mathrm{T}}}{\|\boldsymbol{s}^{\mathrm{T}}\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})\|^2}\\&\times\|\boldsymbol{s}\|\ \|\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})\|(b_0+b_1\|\boldsymbol{q}\|+b_2\|\dot{\boldsymbol{q}}\|^2)\end{aligned} \tag{9.86}$$

稳定性分析：

定义 Lyapunov 函数为

$$V=\frac{1}{2}\boldsymbol{s}^{\mathrm{T}}\boldsymbol{s}$$

由于

$$\begin{aligned}\ddot{\boldsymbol{\varepsilon}}(t)=\ddot{\boldsymbol{q}}-\ddot{\boldsymbol{q}}_r&=\boldsymbol{M}_0^{-1}(\boldsymbol{\tau}+\boldsymbol{\rho}-\boldsymbol{C}_0-\boldsymbol{g}_0)-\ddot{\boldsymbol{q}}_r\\&=\boldsymbol{M}_0^{-1}(\boldsymbol{\tau}_0+\boldsymbol{u}_0+\boldsymbol{u}_1+\boldsymbol{\rho}-\boldsymbol{C}_0-\boldsymbol{g}_0)-\ddot{\boldsymbol{q}}_r\\&=\boldsymbol{M}_0^{-1}(\boldsymbol{C}_0+\boldsymbol{g}_0+\boldsymbol{M}_0\ddot{\boldsymbol{q}}_r+\boldsymbol{u}_0+\boldsymbol{u}_1+\boldsymbol{\rho}-\boldsymbol{C}_0-\boldsymbol{g}_0)-\ddot{\boldsymbol{q}}_r\\&=\boldsymbol{M}_0^{-1}(\boldsymbol{M}_0\ddot{\boldsymbol{q}}_r+\boldsymbol{u}_0+\boldsymbol{u}_1+\boldsymbol{\rho})-\ddot{\boldsymbol{q}}_r=\boldsymbol{M}_0^{-1}(\boldsymbol{u}_0+\boldsymbol{u}_1+\boldsymbol{\rho})\end{aligned}$$

$$\begin{aligned}&\dot{\boldsymbol{\varepsilon}}+\frac{p}{q}\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})\boldsymbol{u}_0(t)\\&\quad=\dot{\boldsymbol{\varepsilon}}+\frac{p}{q}\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})\left(-\frac{q}{p}\boldsymbol{M}_0(\boldsymbol{q})\boldsymbol{C}_1^{-1}\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{2-p/q})\right)\\&\quad=\dot{\boldsymbol{\varepsilon}}-\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{2-p/q})\\&\quad=0\end{aligned}$$

则

$$\begin{aligned}\dot{V}=\boldsymbol{s}^{\mathrm{T}}\dot{\boldsymbol{s}}&=\boldsymbol{s}^{\mathrm{T}}\left(\dot{\boldsymbol{\varepsilon}}+\frac{p}{q}\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\ddot{\boldsymbol{\varepsilon}}\right)\\&=\boldsymbol{s}^{\mathrm{T}}\left\{\dot{\boldsymbol{\varepsilon}}+\frac{p}{q}\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})[\boldsymbol{u}_1(t)+\boldsymbol{u}_0(t)+\boldsymbol{\rho}(t)]\right\}\\&=\boldsymbol{s}^{\mathrm{T}}\left\{\frac{p}{q}\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})[\boldsymbol{u}_1(t)+\boldsymbol{\rho}(t)]\right\}\\&=\boldsymbol{s}^{\mathrm{T}}\left\{\frac{p}{q}\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})\left(-\frac{[\boldsymbol{s}^{\mathrm{T}}\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})]^{\mathrm{T}}}{\|\boldsymbol{s}^{\mathrm{T}}\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})\|^2}\right.\right.\\&\qquad\left.\left.\times\|\boldsymbol{s}\|\ \|\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})\|(b_0+b_1\|\boldsymbol{q}\|+b_2\|\dot{\boldsymbol{q}}\|^2)+\boldsymbol{\rho}(t)\right)\right\}\\&=-\frac{p}{q}\|\boldsymbol{s}\|\ \|\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})\|(b_0+b_1\|\boldsymbol{q}\|+b_2\|\dot{\boldsymbol{q}}\|^2)\\&\qquad+\frac{p}{q}\boldsymbol{s}^{\mathrm{T}}\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})\boldsymbol{\rho}(t)\\&\leqslant-\frac{p}{q}\|\boldsymbol{s}\|\ \|\boldsymbol{C}_1\mathrm{diag}(\dot{\boldsymbol{\varepsilon}}^{p/q-1})\boldsymbol{M}_0^{-1}(\boldsymbol{q})\|(b_0+b_1\|\boldsymbol{q}\|+b_2\|\dot{\boldsymbol{q}}\|^2)\end{aligned}$$

$$+\frac{p}{q}\|s\|\ \|C_1\mathrm{diag}(\dot{\varepsilon}^{p/q-1})M_0^{-1}(q)\|\ \|\rho(t)\|$$

$$=-\frac{p}{q}\|s\|\ \|C_1\mathrm{diag}(\dot{\varepsilon}^{p/q-1})M_0^{-1}(q)\|\ [(b_0+b_1\|q\|+b_2\|\dot{q}\|^2)-\|\rho(t)\|]<0$$

（当 $\|s\|\neq 0$ 时）

9.4.5 仿真实例

二关节刚性机器人模型如图 9-36 所示，其动态方程为

$$M_0(q)\ddot{q}+C_0(q,\dot{q})+g_0(q)=\tau(t)+\rho(t)$$

其中

$$q=\begin{bmatrix}q_1\\q_2\end{bmatrix},\quad \tau=\begin{bmatrix}\tau_1\\\tau_2\end{bmatrix},\quad M_0(q)=\begin{bmatrix}a_{11}(q_2) & a_{12}(q_2)\\a_{12}(q_2) & a_{22}\end{bmatrix},$$

$$C_0(q,\dot{q})=\begin{bmatrix}-\beta_{12}(q_2)\dot{q}_1^2-2\beta_{12}(q_2)\dot{q}_1\dot{q}_2\\\beta_{12}(q_2)\dot{q}_2^2\end{bmatrix},g_0(q)=\begin{bmatrix}\gamma_1(q_1,q_2)g\\\gamma_2(q_1,q_2)g\end{bmatrix},$$

$$\rho(t)=0.1+0.2q+0.3\dot{q}^{\mathrm{T}}\dot{q}$$

取

$$a_{11}(q_2)=(m_1+m_2)r_1^2+m_2r_2^2+2m_2r_1r_2\cos(q_2)+J_1,$$
$$a_{12}(q_2)=m_2r_2^2+m_2r_1r_2\cos(q_2),$$
$$a_{22}=m_2r_2^2+J_2,\beta_{12}(q_2)=m_2r_1r_2\sin(q_2),$$
$$\gamma_1(q_1,q_2)=(m_1+m_2)r_1\cos(q_2)+m_2r_2\cos(q_1+q_2),$$
$$\gamma_2(q_1,q_2)=m_2r_2\cos(q_1+q_2)$$

动态方程的参数为 $r_1=1\mathrm{m}$，$r_2=0.8\mathrm{m}$，$J_1=5\mathrm{kgm}$，$J_2=5\mathrm{kgm}$，$m_1=0.5\mathrm{kg}$，$m_2=1.5\mathrm{kg}$。

二关节的位置指令分别为：$q_{r1}=1.25-(7/5)\mathrm{e}^{-t}+(7/20)\mathrm{e}^{-4t}$，$q_{r2}=1.25+\mathrm{e}^{-t}-(1/4)\mathrm{e}^{-4t}$。系统初始状态条件为 $q_1(0)=1.0$，$q_2(0)=1.5$，$\dot{q}_1(0)=0.0$，$\dot{q}_2(0)=0.0$。滑模控制参数取 $C_1=\begin{bmatrix}c_{11} & 0\\0 & c_{22}\end{bmatrix}=\begin{bmatrix}150 & 0\\0 & 150\end{bmatrix}$，$q=3$，$p=5$，$b_0=1$，$b_1=2$，$b_2=3$。仿真结果如图 9-37～图 9-42 所示。

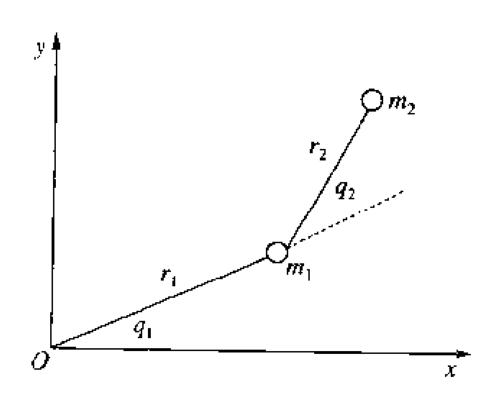


图 9-36 双自由度刚性机器人模型

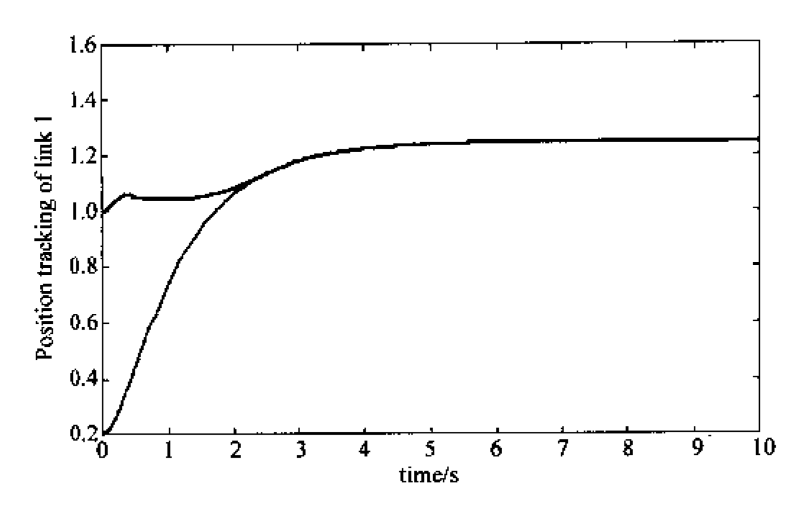


图 9-37 关节 1 的位置跟踪

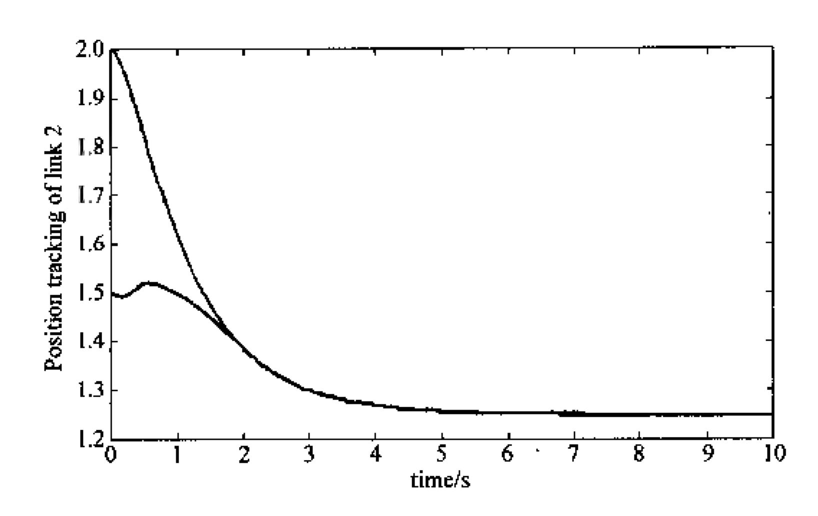


图 9-38 关节 2 的位置指令

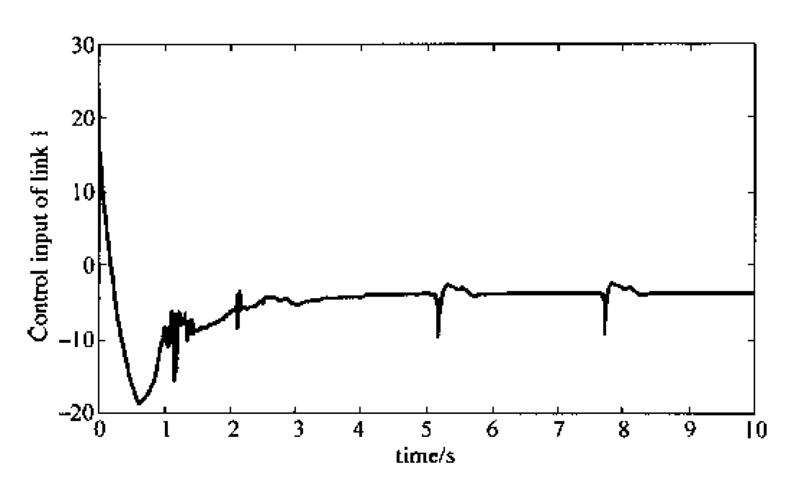


图 9-39 关节 1 的控制输入

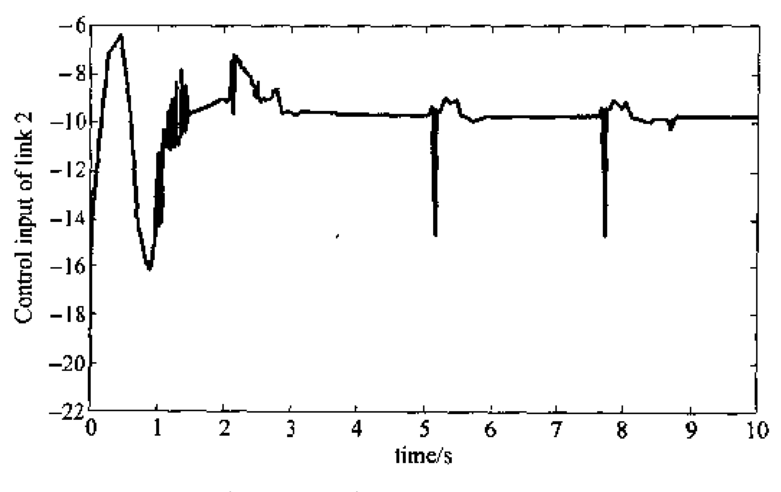


图 9-40 关节 2 的控制输入

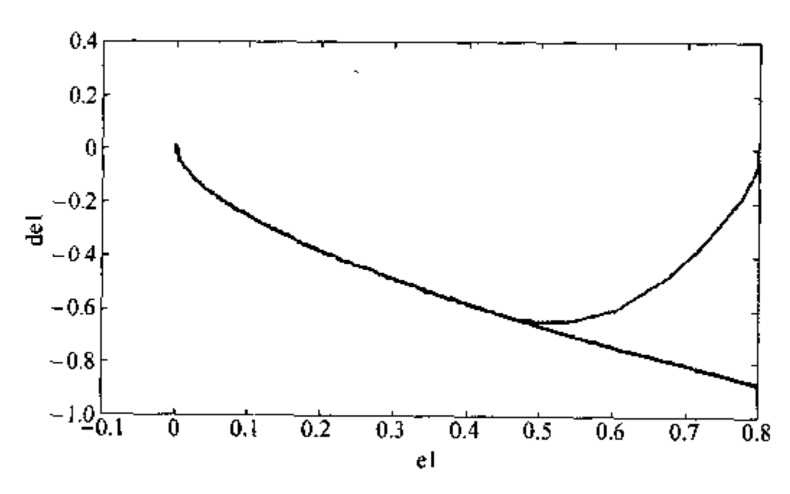


图 9-41 关节 1 的相轨迹

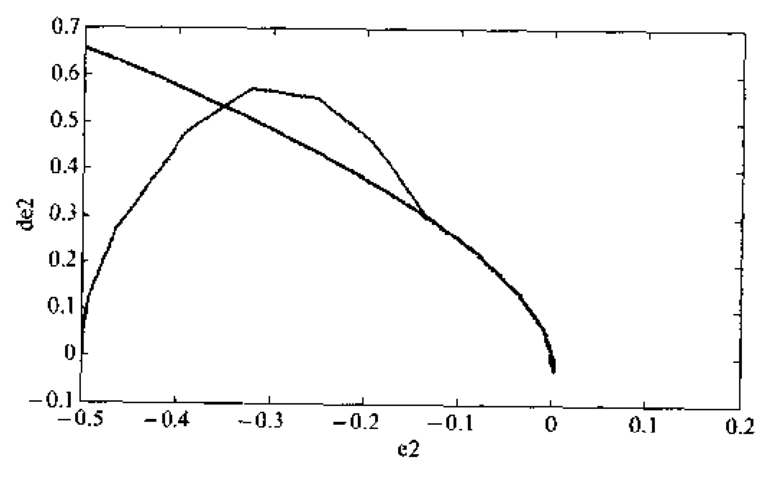


图 9-42 关节 2 的相轨迹

仿真程序：

(1) 主程序(如图 9-43 所示)：chap9_8sim.mdl

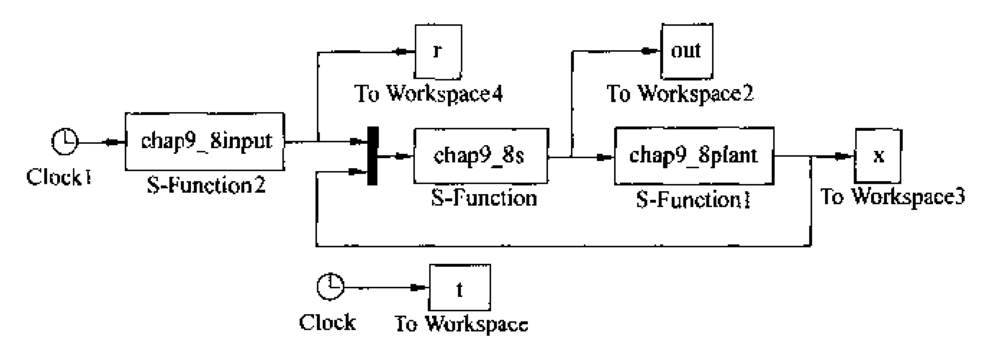


图 9-43 主程序图

(2) S 函数位置指令子程序:chap9_8input.m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];
function sys = mdlOutputs(t,x,u)
qr1 = 1.25 - 7/5 * exp( - t) + 7/20 * exp( - 4 * t);
qr2 = 1.25 + exp( - t) - 1/4 * exp( - 4 * t);

sys(1) = qr1;
```

```
sys(2) = qr2;
```

(3) S 函数控制子程序:chap9_8s. m

```
%S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys = [];
%Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 6;
sizes.NumInputs      = 6;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)
qr1 = u(1);
dqr1 = 7/5 * exp( - t) - 7/5 * exp( - 4 * t);
ddqr1 = - 7/5 * exp( - t) + 28/5 * exp( - 4 * t);

qr2 = u(2);
dqr2 = - exp( - t) + exp( - 4 * t);
ddqr2 = exp( - t) - 4 * exp( - 4 * t);

x = u(3:1:6);
e1 = x(1) - qr1;
e2 = x(3) - qr2;
e = [e1;e2];
de1 = x(2) - dqr1;
```

```
de2 = x(4) - dqr2;
de = [de1;de2];

q = [x(1);x(3)];
dq = [x(2);x(4)];

s1 = e1 + (abs(de1))^(5/3) * sign(de1);
s2 = e2 + (abs(de2))^(5/3) * sign(de2);
s = [s1;s2];

r1 = 1;r2 = 0.8;
J1 = 5;J2 = 5;
m1 = 0.5;m2 = 1.5;
g = 9.8;

a11 = (m1 + m2) * r1^2 + m2 * r2^2 + 2 * m2 * r1 * r2 * cos(x(3)) + J1;
a12 = m2 * r1^2 + m2 * r1 * r2 * cos(x(3));
a22 = m2 * r2^2 + J2;
b12 = m2 * r1 * r2 * sin(x(3));

gama1 = ((m1 + m2) * r1 * cos(x(3)) + m2 * r2 * cos(x(1) + x(3)));
gama2 = m2 * r2 * cos(x(1) + x(3));

M0 = [a11 a12;a12 a22];
C0 = [ - b12 * x(2) * x(2) - 2 * b12 * x(2) * x(4);b12 * x(4) * x(4)];
g0 = [gama1 * g;gama2 * g];

ddqr = [ddqr1;ddqr2];
tao0 = C0 - g0 + M0 * ddqr;

z11 = (abs(de1))^(1/3) * sign(de1);
z12 = (abs(de2))^(1/3) * sign(de2);

z21 = (abs(de1))^(2/3);
z22 = (abs(de2))^(2/3);

C1 = [150 0;0 150];
u0 = - 0.6 * M0 * inv(C1) * [z11;z12];

u2 = - (s' * C1 * [z21 0;0 z22] * inv(M0))'/(norm(s' * C1 * [z21 0;0 z22] * inv(M0)) + 0.001)^2;
b0 = 1;b1 = 2;b2 = 3;
u1 = u2 * [norm(s) * norm(C1 * [z21 0;0 z22] * inv(M0)) * (b0 + b1 * norm(q) + b2 * norm(dq) * norm
(dq))];

tao = tao0 + u0 + u1;
tao1 = tao(1);
tao2 = tao(2);

sys(1) = tao1;
sys(2) = tao2;
sys(3) = e1;
```

```
sys(4) = de1;
sys(5) = e2;
sys(6) = de2;
```

(4) S 函数被控对象子程序:chap9_8plant. m

```
% S-function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

% mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 4;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 4;
sizes.NumInputs      = 6;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [1.0,0,1.5,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)

r1 = 1;r2 = 0.8;
J1 = 5;J2 = 5;
m1 = 0.5;m2 = 1.5;
g = 9.8;

a11 = (m1 + m2) * r1^2 + m2 * r2^2 + 2 * m2 * r1 * r2 * cos(x(3)) + J1;
a12 = m2 * r1^2 + m2 * r1 * r2 * cos(x(3));
a22 = m2 * r2^2 + J2;
```

```
b12 = m2 * r1 * r2 * sin(x(3));

gama1 = ((m1 + m2) * r1 * cos(x(3)) + m2 * r2 * cos(x(1) + x(3)));
gama2 = m2 * r2 * cos(x(1) + x(3));

M0 = [a11 a12;a12 a22];
C0 = [ - b12 * x(2) * x(2) - 2 * b12 * x(2) * x(4);b12 * x(4) * x(4)];
g0 = [gama1 * g;gama2 * g];

tao = u(1:2);

q = [x(1);x(3)];
dq = [x(2);x(4)];
rou = 0.1 - 0.2 * q + 0.3 * dq' * dq;

Y = inv(M0) * (tao + rou - C0 - g0);

sys(1) = x(2);
sys(2) = Y(1);
sys(3) = x(4);
sys(4) = Y(2);

function sys = mdlOutputs(t,x,u)

sys(1) = x(1);
sys(2) = x(2);
sys(3) = x(3);
sys(4) = x(4);
```

(5) 作图子程序:chap9_8plot.m

```
close all;

figure(1);
plot(t,r(:,1),'r',t,x(:,1),'b');
xlabel('time(s)');ylabel('Position tracking of link1');

figure(2);
plot(t,r(:,2),'r',t,x(:,3),'b');
xlabel('time(s)');ylabel('Position tracking of link2');

tao1 = out(:,1);
tao2 = out(:,2);
figure(3);
plot(t,tao1,'r');
xlabel('time(s)');ylabel('Control input of link1');

figure(4);
plot(t,tao2,'r');
xlabel('time(s)');ylabel('Control input of link2');
```

```
e1 = out(:,3);
de1 = out(:,4);
e2 = out(:,5);
de2 = out(:,6);

q = 3;p = 5;
figure(5);
plot(e1,de1,'r',e1,(abs(-e1)).^(q/p).*sign(-e1),'b');
xlabel('e1');ylabel('de1');

figure(6);
plot(e2,de2,'r',e2,(abs(-e2)).^(q/p).*sign(-e2),'b');
xlabel('e2');ylabel('de2');
```

参考文献

1. 庄开宇，张克勤，苏宏业，褚健. 高阶非线性系统的 Terminal 滑模控制. 浙江大学学报，2002，36(5)：482～485
2. Park K B. Tsuiji T. Terminal sliding mode control of second-order nonlinear uncertain systems. International Journal of Robust and Nonlinear Control, 1999, 9 (11): 769～780
3. Feng Y, Yu X H, Man Z H. Non-singular terminal sliding mode control of rigid manipulators, Automatica. 2002, 38: 2159～2167
4. Yu S H, Yu X H, Man Z H. Robust Global Terminal Sliding Mode Control Of SISO Nonlinear Uncertain Systems [A]. Proceedings of 39th IEEE Conference on Decision and Control, Sydney, Australia, December, 2000: 2198～2203
5. Yu X, Man Z. Fast terminal sliding mode control for single input systems. Proceedings of 2000 Asian Control Conference, Shanghai, China, 2000

第 10 章　几种新型滑模控制

10.1　二阶不确定系统的全局滑模控制

传统的滑模变结构控制系统的响应包括趋近模态和滑动模态两部分，该类系统对系统参数的不确定性和外部扰动的鲁棒性仅存在于滑动模态阶段，系统的动力学特性在响应的全过程并不具有鲁棒性。

全局滑模控制是通过设计一种动态非线性滑模面方程来实现的。全局滑模控制消除滑模控制的到达运动阶段，使系统在响应的全过程都具有鲁棒性，克服了传统滑模变结构控制中到达模态不具有鲁棒性的特点。

10.1.1　系统描述

被控对象为 SISO 二阶不确定系统：

$$\ddot{x} + a(t)\dot{x} = b(t)[u + d(t)] \tag{10.1}$$

其中 $b(t)>0$，$d(t)$为外部干扰。假设

$$\beta_{\min} \leqslant b^{-1}(t) \leqslant \beta_{\max} \tag{10.2}$$

$$\alpha_{\min} \leqslant b^{-1}(t)a(t) \leqslant \alpha_{\max} \tag{10.3}$$

$$d(t) < D \tag{10.4}$$

10.1.2　全局滑模切换函数设计

设位置指令为 r，则误差为

$$e = x - r \tag{10.5}$$

全局动态滑模面设计为

$$s = \dot{e} + ce - f(t) \tag{10.6}$$

其中 $f(t)$是为了达到全局滑模面设计的函数，$f(t)$满足以下三个条件：

(1) $f(0)=\dot{e}_0+ce_0$；

(2) $t\to\infty$时，$f(t)\to 0$；

(3) $f(t)$具有一阶导数。

根据上述三个条件，可将 $f(t)$设计为

$$f(t) = f(0)e^{-kt} \tag{10.7}$$

10.1.3 控制器设计

全局滑模控制律设计为

$$u=-\hat{\beta}(c\dot{x}-\dot{f})+\hat{\alpha}\dot{x}-\hat{\beta}R-\{\Delta\beta\,|c\dot{x}-\dot{f}|+\Delta\alpha\,|\dot{x}|+D+\Delta\beta\,|R|\}\mathrm{sgn}(s) \tag{10.8}$$

其中

$$\hat{\beta}=\frac{\beta_{\max}+\beta_{\min}}{2},\Delta\beta=\frac{\beta_{\max}-\beta_{\min}}{2} \tag{10.9}$$

$$\hat{\alpha}=\frac{\alpha_{\max}+\alpha_{\min}}{2},\Delta\alpha=\frac{\alpha_{\max}-\alpha_{\min}}{2} \tag{10.10}$$

10.1.4 稳定性分析

定义 Lyapunov 函数为

$$V=\frac{1}{2}s^2 \tag{10.11}$$

由式(10.6)得

$$\begin{aligned}\dot{s}&=\ddot{e}+c\dot{e}-\dot{f}=\ddot{x}-\ddot{r}+c(\dot{x}-\dot{r})-\dot{f}\\&=\ddot{x}+(c\dot{x}-\dot{f})-(\ddot{r}+c\dot{r})\\&=\ddot{x}+(c\dot{x}-\dot{f})+R=-a\dot{x}+bu+bd+(c\dot{x}-\dot{f})+R\\&=b[b^{-1}(c\dot{x}-\dot{f})+b^{-1}R-b^{-1}a\dot{x}+u+d]\end{aligned}$$

其中 $R=-(\ddot{r}+c\dot{r})$。

将控制律式(10.8)代入上式，得

$$\begin{aligned}b^{-1}\dot{s}=&\,b^{-1}(c\dot{x}-\dot{f})+b^{-1}R-b^{-1}a\dot{x}-\hat{\beta}(c\dot{x}-\dot{f})+\hat{\alpha}\dot{x}-\hat{\beta}R\\&-\{\Delta\beta\,|c\dot{x}-\dot{f}|+\Delta\alpha\,|\dot{x}|+D+\Delta\beta\,|R|\}\mathrm{sgn}(s)+d\\=&\,(b^{-1}-\hat{\beta})(c\dot{x}-\dot{f})-\Delta\beta\,|c\dot{x}-\dot{f}|\,\mathrm{sgn}(s)+(b^{-1}-\hat{\beta})R\\&-\Delta\beta\,|R|\,\mathrm{sgn}(s)+(\hat{\alpha}-b^{-1}a)\dot{x}-\Delta\alpha\,|\dot{x}|\,\mathrm{sgn}(s)+d-D\mathrm{sgn}(s)\end{aligned}$$

则

$$\begin{aligned}b^{-1}\dot{V}=b^{-1}s\dot{s}=&\,(b^{-1}-\hat{\beta})(c\dot{x}-\dot{f})s-\Delta\beta\,|c\dot{x}-\dot{f}|\,|s|+(b^{-1}-\hat{\beta})Rs\\&-\Delta\beta\,|R|\,|s|+(\hat{\alpha}-b^{-1}a)\dot{x}s-\Delta\alpha\,|\dot{x}|\,|s|+ds-D\,|s|\end{aligned}$$

由式(10.2)～式(10.4)及式(10.9)和式(10.10)可知

$$b^{-1}-\hat{\beta}\leqslant\Delta\beta$$
$$\hat{\alpha}-b^{-1}a\leqslant\Delta\alpha$$

即

$$\dot{V}\leqslant 0 \tag{10.12}$$

为了消除抖振，可采用饱和函数方法，即用 sat(s)代替 sgn(s)。

$$\mathrm{sat}(s)=\begin{cases}1 & s/\varphi>1\\ \dfrac{s}{\varphi} & |s/\varphi|\leqslant 1\\ -1 & s/\varphi<-1\end{cases} \tag{10.13}$$

其中 φ 为很小的正常数。

10.1.5 仿真实例

被控对象为

$$\ddot{x}+a(t)\dot{x}=b(t)[u+d(t)]$$

其中 $a(t)=25+5\sin(t)$，$b(t)=133+10\sin(t)$，$d(t)=0.1\sin(2\pi t)$。

由已知可得：$a_{\min}=20$，$a_{\max}=30$，$b_{\min}=123$，$b_{\max}=143$，$D=0.10$。由式(10.2)和式(10.3)可得 $\alpha_{\min}=0.1399$，$\alpha_{\max}=0.2439$，$\beta_{\min}=0.0070$ 及 $\beta_{\max}=0.0081$，由式(10.9)和式(10.10)可得$\hat{\beta}=0.0076$，$\Delta\beta=5.6854\mathrm{e}-004$，$\hat{\alpha}=0.1919$ 及 $\Delta\alpha=0.0520$。

位置指令取 $r=0.5\sin(2\pi t)$。采用控制律式(10.8)，取 $c=10$，将 $f(t)$ 设计为 $f(t)=s(0)\mathrm{e}^{-130t}$。$M=1$ 和 $M=2$ 分别为采用饱和函数和切换函数。取 $M=2$，$\varphi=0.05$，仿真结果如图 10-1～图 10-3 所示。由图可见，采用饱和函数有效地降低了抖振。

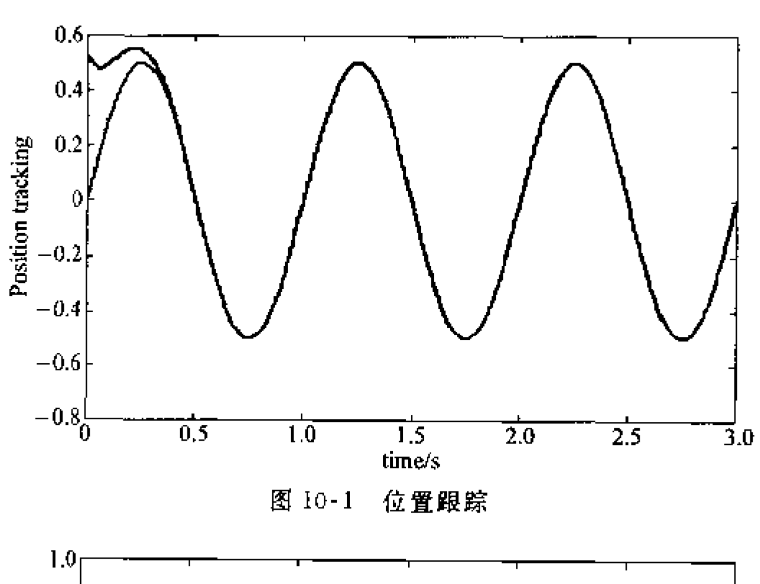


图 10-1　位置跟踪

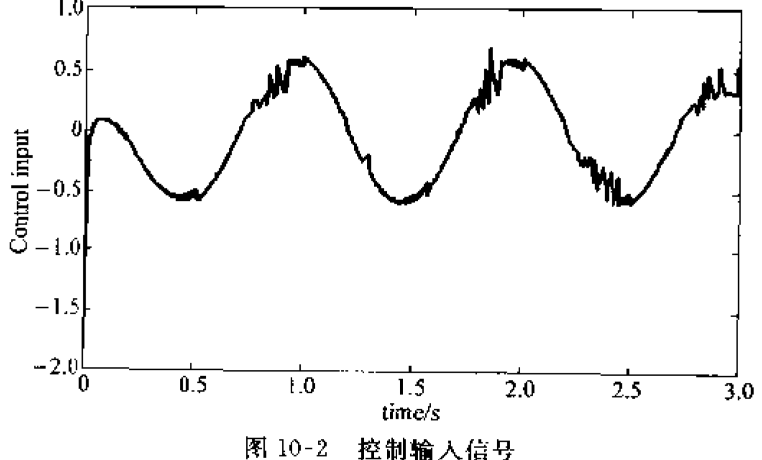


图 10-2　控制输入信号

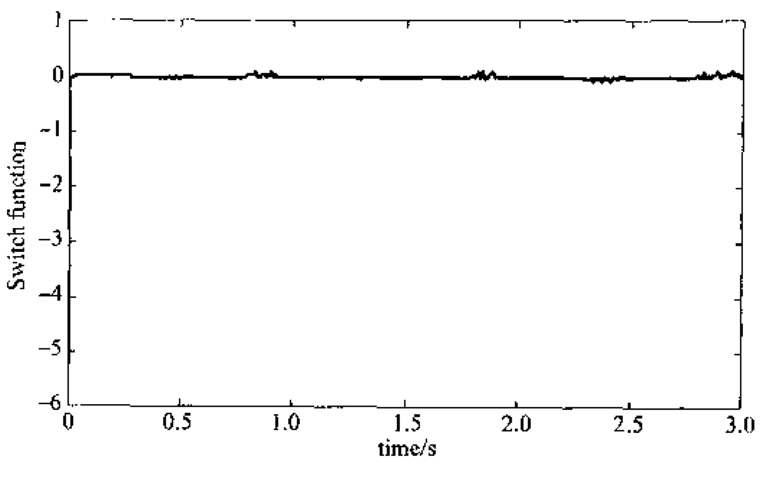


图 10-3 切换函数

仿真程序：

（1）Simulink 主程序（如图 10-4 所示）：chap10_1sim. mdl

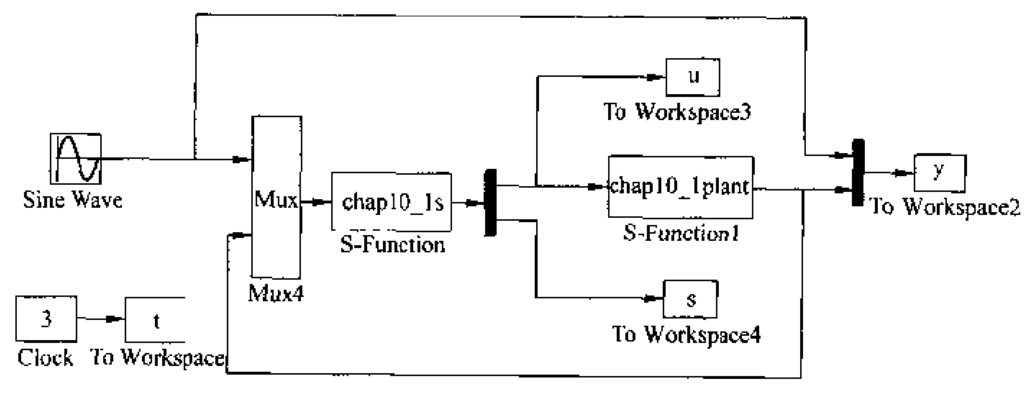


图 10-4 主程序图

（2）控制器 S 函数程序：chap10_1s. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 0;
sizes.NumDiscStates   = 0;
```

```
sizes.NumOutputs      = 2;
sizes.NumInputs       = 3;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 1;
sys = simsizes(sizes);
x0   = [];
str  = [];
ts   = [0 0];

function sys = mdlOutputs(t,x,u)
r = u(1);
dr = 0.5 * 2 * pi * cos(2 * pi * t);
ddr = -0.5 * (2 * pi)^2 * sin(2 * pi * t);

c = 10;
e = u(2) - r;
de = u(3) - dr;

dt = 0.10 * sin(2 * pi * t);
D = 0.10;

e0 = pi/6;
de0 = 0 - 0.5 * 2 * pi;
s0 = de0 + c * e0;
ft = s0 * exp(-130 * t);
df = -130 * s0 * exp(-130 * t);

s = de + c * e - ft;
R = -(ddr + c * dr);

a = 25 + 5 * sin(t);
amin = 20;amax = 30;
b = 133 + 10 * sin(t);
bmin = 123;bmax = 143;

beta_min = 1/bmax;
beta_max = 1/bmin;
beta_p = (beta_min + beta_max)/2;
beta_d = (beta_max - beta_min)/2;

alfa_min = amin/bmax;
alfa_max = amax/bmin;
alfa_p = (alfa_min + alfa_max)/2;
alfa_d = (alfa_max - alfa_min)/2;

M = 2;
if M == 1
ut = -beta_p * (c * u(3) - df) + alfa_p * u(3) - beta_p * R - ...
    [beta_d * abs(c * u(3) - df) + alfa_d * abs(u(3)) + D + beta_d * abs(R)] * sign(s);
elseif M == 2
    fai = 0.05;
```

```
    if s/fai>1
      sat = 1;
    elseif abs(s/fai)< = 1
      sat = s/fai;
    elseif s/fai< - 1
      sat = - 1;
    end
ut = - beta_p * (c * u(3) - df) + alfa_p * u(3) - beta_p * R - ...
   [beta_d * abs(c * u(3) - df) + alfa_d * abs(u(3)) + D + beta_d * abs(R)] * sat;
end
sys(1) = ut;
sys(2) = s;
```

(3) 被控对象 S 函数程序：chap10_1plant. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;
sys = simsizes(sizes);
x0 = [pi/6;0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
a = 25 + 5 * sin(t);
b = 133 + 10 * sin(t);
dt = 0.10 * sin(2 * pi * t);

sys(1) = x(2);
sys(2) = - a * x(2) + b * (u + dt);
function sys = mdlOutputs(t,x,u)
```

```
sys(1) = x(1);
sys(2) = x(2);
```

(4) 作图程序:chap10_1plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,s(:,1),'r');
xlabel('time(s)');ylabel('Switch function');
```

10.2　N阶不确定系统的全局滑模控制

10.2.1　系统描述

被控对象为一 n 阶 SISO 不确定系统:

$$\begin{cases}\dot{x}_i = x_{i+1} \\ \dot{x}_n = \phi(x,t)\boldsymbol{x} + b(\boldsymbol{x},t)u + d(\boldsymbol{x},t)\end{cases} \tag{10.14}$$

其中 $i=1,2,\cdots,n-1$,$\phi(x,t)=[\phi_1(\boldsymbol{x},t)\ \phi_2(\boldsymbol{x},t)\ \cdots\ \phi_n(\boldsymbol{x},t)]$,$\boldsymbol{x}=[x_1\ x_2\cdots\ x_n]^{\mathrm{T}}$,$d(\boldsymbol{x},t)$为外部干扰。

假设1:ϕ 和 $d(\boldsymbol{x},t)$满足

$$\phi_{i\min} \leqslant \phi_i(\boldsymbol{x},t) \leqslant \phi_{i\max}, i=1,2,\cdots,n \tag{10.15}$$

$$d_{\min} \leqslant d(\boldsymbol{x},t) \leqslant d_{\max} \tag{10.16}$$

假设2:$b(\boldsymbol{x},t)$满足

$$b_{\min} \leqslant b(\boldsymbol{x},t) \leqslant b_{\max}, b(\boldsymbol{x},t) > 0 \tag{10.17}$$

10.2.2　控制器设计

设位置指令为

$$\begin{cases}x_{r1} = r \\ \dot{x}_{ri} = x_{ri+1}\end{cases} \tag{10.18}$$

其中 $i=1,2,\cdots,n$。

全局动态滑模面设计为

$$\sigma(\boldsymbol{x}) = \sigma_1(\boldsymbol{x}) - \sigma_0(\boldsymbol{x})\mathrm{e}^{-\lambda t} \tag{10.19}$$

其中 λ 为一个正的常数，c_i 满足 Hurwitz 条件，$c_n=1$。

定义

$$\sigma_1(\boldsymbol{x}) = \sum_{i=1}^{n} c_i(x_{ri} - x_i) \tag{10.20}$$

$$\sigma_0(\boldsymbol{x}) = \sum_{i=1}^{n} c_i(x_{ri0} - x_{i0}) \tag{10.21}$$

其中 x_{ri0} 和 x_{i0} 分别为 x_{ri}，x_i 的初始值。

全局滑模控制律为

$$u = u_c + u_{vss} \tag{10.22}$$

其中 u_c 为线性控制项，u_{vss} 为切换项。

线性控制律为

$$u_c = \left[k\sigma + r^{(n)} - \bar{\phi}\boldsymbol{x} - \bar{d} + \sum_{i=1}^{n-1} c_i(x_{ri+1} - x_{i+1})\right]/b_{\min} \tag{10.23}$$

其中 k 为正的常数。$\bar{\phi}$ 和 $\bar{d}(\boldsymbol{x},t)$ 满足

$$\bar{\phi} = \frac{1}{2}(\phi_{\min} + \phi_{\max}), \bar{d}(\boldsymbol{x},t) = \frac{1}{2}(d_{\min} + d_{\max}) \tag{10.24}$$

控制律中的切换项设计为

$$u_{vss} = (\varepsilon + |\lambda\sigma_0 e^{-\lambda t}|/b_{\min})\operatorname{sgn}(\sigma) \tag{10.25}$$

式中增益项 ε 为

$$\varepsilon \geqslant (\Delta\tilde{b}\,|r^{(n)}| + \tilde{\phi}\,|\boldsymbol{x}|_1 + \tilde{d} + \Delta\tilde{b}\sum_{i=1}^{n-1} c_i\,|x_{ri+1} - x_{i+1}|)/b_{\min} \tag{10.26}$$

其中 $|\boldsymbol{x}|_1=[|x_1| \quad |x_2| \quad \cdots \quad |x_n|]^{\mathrm{T}}$，$\Delta\tilde{b}=\frac{b_{\max}}{b_{\min}}-1$，$\tilde{\phi}=\frac{1}{2}(\phi_{\max}-\phi_{\min})$，$\tilde{d}=\frac{1}{2}(d_{\max}-d_{\min})$。

10.2.3 稳定性分析

定义 Lyapunov 函数为

$$V = \frac{1}{2}\sigma^2 \tag{10.27}$$

则

$$\dot{V} = \sigma\dot{\sigma} \tag{10.28}$$

$$\begin{aligned}
\dot{\sigma} &= \dot{\sigma}_1 + \lambda\sigma_0 e^{-\lambda t} \\
&= c_n(\dot{x}_{rn} - \dot{x}_n) + \sum_{i=1}^{n-1} c_i(x_{ri+1} - x_{i+1}) + \lambda\sigma_0 e^{-\lambda t} \\
&= r^{(n)} - \phi\boldsymbol{x} - b(\boldsymbol{x},t)u - d(\boldsymbol{x},t) + \sum_{i=1}^{n-1} c_i(x_{ri+1} - x_{i+1}) + \lambda\sigma_0 e^{-\lambda t} \\
&= r^{(n)} - \phi\boldsymbol{x} - d(\boldsymbol{x},t) + \sum_{i=1}^{n-1} c_i(x_{ri+1} - x_{i+1}) + \lambda\sigma_0 e^{-\lambda t} \\
&\quad + \left[-\frac{b(\boldsymbol{x},t)}{b_{\min}}k\sigma - \frac{b(\boldsymbol{x},t)}{b_{\min}}r^{(n)} + \frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{\phi}\boldsymbol{x} + \frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{d} - \frac{b(\boldsymbol{x},t)}{b_{\min}}\right.
\end{aligned}$$

$$
\begin{aligned}
&\times \sum_{i=1}^{n-1} c_i(x_{ri+1} - x_{i+1}) - \frac{b(\boldsymbol{x},t)}{b_{\min}} \mid \lambda\sigma_0 \mathrm{e}^{-\lambda t} \mid \operatorname{sgn}(\sigma) - b(\boldsymbol{x},t)\varepsilon \operatorname{sgn}(\sigma) \Big] \\
&= \left(1 - \frac{b(\boldsymbol{x},t)}{b_{\min}}\right) r^{(n)} + \left(\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{\phi} - \phi\right)\boldsymbol{x} + \left[\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{d} - d(\boldsymbol{x},t)\right] \\
&\quad - \frac{b(\boldsymbol{x},t)}{b_{\min}} k\sigma + \left[1 - \frac{b(\boldsymbol{x},t)}{b_{\min}}\right] \sum_{i=1}^{n-1} c_i(x_{ri+1} - x_{i+1}) \\
&\quad - \frac{b(\boldsymbol{x},t)}{b_{\min}} \mid \lambda\sigma_0 \mathrm{e}^{-\lambda t} \mid \operatorname{sgn}(\sigma) - b(\boldsymbol{x},t)\varepsilon \operatorname{sgn}(\sigma) + \lambda\sigma_0 \mathrm{e}^{-\lambda t}
\end{aligned} \tag{10.29}
$$

由于

$$
\lambda\sigma\sigma_0 \mathrm{e}^{-\lambda t} \leqslant \frac{b(\boldsymbol{x},t)}{b_{\min}} \mid \lambda\sigma\sigma_0 \mathrm{e}^{-\lambda t} \mid \tag{10.30}
$$

由式(10.17)和式(10.26)得

$$
-b(\boldsymbol{x},t)\varepsilon \leqslant -\frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\Delta\tilde{b} \mid r^{(n)} \mid + \tilde{\phi} \mid \boldsymbol{x} \mid_1 + \tilde{d} + \Delta\tilde{b}\sum_{i=1}^{n-1} c_i \mid x_{ri+1} - x_{i+1} \mid\right) \tag{10.31}
$$

则

$$
\begin{aligned}
\sigma\dot{\sigma} &= \left(1 - \frac{b(\boldsymbol{x},t)}{b_{\min}}\right) r^{(n)}\sigma + \left(\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{\phi} - \phi\right)\boldsymbol{x}\sigma + \left(\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{d} - d(\boldsymbol{x},t)\right)\sigma \\
&\quad - \frac{b(\boldsymbol{x},t)}{b_{\min}} k\sigma^2 + \left[1 - \frac{b(\boldsymbol{x},t)}{b_{\min}}\right] \sum_{i=1}^{n-1} c_i(x_{ri+1} - x_{i+1})\sigma \\
&\quad - \frac{b(\boldsymbol{x},t)}{b_{\min}} \mid \lambda\sigma\sigma_0 \mathrm{e}^{-\lambda t} \mid - b(\boldsymbol{x},t)\varepsilon \mid \sigma \mid + \lambda\sigma\sigma_0 \mathrm{e}^{-\lambda t} \\
&\leqslant \left(1 - \frac{b(\boldsymbol{x},t)}{b_{\min}}\right) r^{(n)}\sigma + \left(\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{\phi} - \phi\right)\boldsymbol{x}\sigma + \left[\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{d} - d(\boldsymbol{x},t)\right]\sigma \\
&\quad - \frac{b(\boldsymbol{x},t)}{b_{\min}} k\sigma^2 + \left(1 - \frac{b(\boldsymbol{x},t)}{b_{\min}}\right) \sum_{i=1}^{n-1} c_i(x_{ri+1} - x_{i+1})\sigma \\
&\quad - \frac{b(\boldsymbol{x},t)}{b_{\min}}\left[\Delta\tilde{b} \mid r^{(n)} \mid + \tilde{\phi} \mid \boldsymbol{x} \mid_1 + \tilde{d} + \Delta\tilde{b}\sum_{i=1}^{n-1} c_i \mid x_{ri+1} - x_{i+1} \mid\right] \mid \sigma \mid \\
&= \left(1 - \frac{b(\boldsymbol{x},t)}{b_{\min}}\right) r^{(n)}\sigma + \left(\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{\phi} - \phi\right)\boldsymbol{x}\sigma + \left[\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{d} - d(\boldsymbol{x},t)\right]\sigma \\
&\quad - \frac{b(\boldsymbol{x},t)}{b_{\min}} k\sigma^2 + \left[1 - \frac{b(\boldsymbol{x},t)}{b_{\min}}\right] \sum_{i=1}^{n-1} c_i(x_{ri+1} - x_{i+1})\sigma \\
&\quad - \frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}} - 1\right) \mid r^{(n)} \mid \mid \sigma \mid - \frac{b(\boldsymbol{x},t)}{b_{\min}} \frac{1}{2}(\phi_{\max} - \phi_{\min}) \mid \boldsymbol{x} \mid_1 \mid \sigma \mid \\
&\quad - \frac{b(\boldsymbol{x},t)}{b_{\min}} \frac{1}{2}(d_{\max} - d_{\min}) \mid \sigma \mid \\
&\quad - \frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}} - 1\right) \sum_{i=1}^{n-1} c_i \mid x_{ri+1} - x_{i+1} \mid \mid \sigma \mid \\
&= \left[\left(1 - \frac{b(\boldsymbol{x},t)}{b_{\min}}\right) r^{(n)}\sigma - \frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}} - 1\right) \mid r^{(n)} \mid \mid \sigma \mid\right] \\
&\quad + \left[\left(1 - \frac{b(\boldsymbol{x},t)}{b_{\min}}\right) \sum_{i=1}^{n-1} c_i(x_{ri+1} - x_{i+1})\right]\sigma
\end{aligned}
$$

$$-\frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}}-1\right)\sum_{i=1}^{n-1}c_i\mid x_{ri+1}-x_{i+1}\mid\mid\sigma\mid$$
$$+\left[\left(\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{d}-d(\boldsymbol{x},t)\right)\sigma-\frac{b(\boldsymbol{x},t)}{b_{\min}}\frac{1}{2}(d_{\max}-d_{\min})\mid\sigma\mid\right]$$
$$+\left[\left(\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{\phi}-\phi\right)\boldsymbol{x}\sigma-\frac{b(\boldsymbol{x},t)}{b_{\min}}\frac{1}{2}(\phi_{\max}-\phi_{\min})\mid\boldsymbol{x}\mid_1\mid\sigma\mid\right]-\frac{b(\boldsymbol{x},t)}{b_{\min}}k\sigma^2$$
$$=S1+S2+S3+S4-\frac{b(\boldsymbol{x},t)}{b_{\min}}k\sigma^2 \tag{10.32}$$

其中

$$S1=\left(1-\frac{b(\boldsymbol{x},t)}{b_{\min}}\right)r^{(n)}\sigma-\frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}}-1\right)\mid r^{(n)}\mid\mid\sigma\mid$$
$$S2=\left(1-\frac{b(\boldsymbol{x},t)}{b_{\min}}\right)\sum_{i=1}^{n-1}c_i(x_{ri+1}-x_{i+1})\sigma$$
$$-\frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}}-1\right)\sum_{i=1}^{n-1}c_i\mid x_{ri+1}-x_{i+1}\mid\mid\sigma\mid \tag{10.33}$$
$$S3=\left(\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{d}-d(\boldsymbol{x},t)\right)\sigma-\frac{b(\boldsymbol{x},t)}{b_{\min}}\frac{1}{2}(d_{\max}-d_{\min})\mid\sigma\mid$$
$$S4=\left(\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{\phi}-\phi\right)\boldsymbol{x}\sigma-\frac{b(\boldsymbol{x},t)}{b_{\min}}\frac{1}{2}(\phi_{\max}-\phi_{\min})\mid\boldsymbol{x}\mid_1\mid\sigma\mid$$

取

$$\phi_{i\min}\leqslant 0,\phi_{i\max}\geqslant 0,i=1,2,\cdots,n \tag{10.34}$$
$$d_{\min}\leqslant 0,d_{\max}\geqslant 0$$

分以下三种情况进行讨论。

情况 1：

由于$\frac{b(\boldsymbol{x},t)}{b_{\min}}-1\leqslant\frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}}-1\right)$，则

$$S1=\left(1-\frac{b(\boldsymbol{x},t)}{b_{\min}}\right)r^{(n)}\sigma-\frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}}-1\right)\mid r^{(n)}\mid\mid\sigma\mid$$
$$\leqslant\left|1-\frac{b(\boldsymbol{x},t)}{b_{\min}}\right|\mid r^{(n)}\mid\mid\sigma\mid-\frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}}-1\right)\mid r^{(n)}\mid\mid\sigma\mid$$
$$=\left(\frac{b(\boldsymbol{x},t)}{b_{\min}}-1\right)\mid r^{(n)}\mid\mid\sigma\mid-\frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}}-1\right)\mid r^{(n)}\mid\mid\sigma\mid\leqslant 0 \tag{10.35}$$
$$S2=\left(1-\frac{b(\boldsymbol{x},t)}{b_{\min}}\right)\sum_{i=1}^{n-1}c_i(x_{ri+1}-x_{i+1})\sigma$$
$$-\frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}}-1\right)\sum_{i=1}^{n-1}c_i\mid x_{ri+1}-x_{i+1}\mid\mid\sigma\mid$$
$$\leqslant\left(\frac{b(\boldsymbol{x},t)}{b_{\min}}-1\right)\left|\sum_{i=1}^{n-1}c_i(x_{ri+1}-x_{i+1})\right|\mid\sigma\mid$$
$$-\frac{b(\boldsymbol{x},t)}{b_{\min}}\left(\frac{b_{\max}}{b_{\min}}-1\right)\sum_{i=1}^{n-1}c_i\mid x_{ri+1}-x_{i+1}\mid\mid\sigma\mid\leqslant 0 \tag{10.36}$$

情况 2：

$$S3=\left[\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{d}-d(\boldsymbol{x},t)\right]\sigma-\frac{b(\boldsymbol{x},t)}{b_{\min}}\frac{1}{2}(d_{\max}-d_{\min})\,|\,\sigma\,| \tag{10.37}$$

当 $\sigma>0$ 时，可得

$$\begin{aligned}S3&=\left[\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{d}-d(\boldsymbol{x},t)\right]\sigma-\frac{b(\boldsymbol{x},t)}{b_{\min}}\frac{1}{2}(d_{\max}-d_{\min})\sigma\\&=\left[\frac{b(\boldsymbol{x},t)}{b_{\min}}d_{\min}-d(\boldsymbol{x},t)\right]\sigma\end{aligned} \tag{10.38}$$

由于 $d_{\min}\leqslant 0,\dfrac{b(\boldsymbol{x},t)}{b_{\min}}\geqslant 1$，则

$$\frac{b(\boldsymbol{x},t)}{b_{\min}}d_{\min}\leqslant d(\boldsymbol{x},t),S3\leqslant 0 \tag{10.39}$$

当 $\sigma<0$ 时，可得

$$\begin{aligned}S3&=\left[\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{d}-d(\boldsymbol{x},t)\right]\sigma+\frac{b(\boldsymbol{x},t)}{b_{\min}}\frac{1}{2}(d_{\max}-d_{\min})\sigma\\&=\left[\frac{b(\boldsymbol{x},t)}{b_{\min}}d_{\max}-d(\boldsymbol{x},t)\right]\sigma\end{aligned} \tag{10.40}$$

由于 $d_{\max}\geqslant 0,\dfrac{b(\boldsymbol{x},t)}{b_{\min}}\geqslant 1$，则

$$\frac{b(\boldsymbol{x},t)}{b_{\min}}d_{\max}\geqslant d(\boldsymbol{x},t),S3\leqslant 0 \tag{10.41}$$

情况 3：

$$S4=\left(\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{\phi}-\phi\right)\boldsymbol{x}\sigma-\frac{b(\boldsymbol{x},t)}{b_{\min}}\frac{1}{2}(\phi_{\max}-\phi_{\min})\,|\,\boldsymbol{x}\,|_1\,|\,\sigma\,| \tag{10.42}$$

令

$$S4_i=\left(\frac{b(\boldsymbol{x},t)}{b_{\min}}\bar{\phi}_i-\phi_i\right)x_i\sigma-\frac{b(\boldsymbol{x},t)}{b_{\min}}\frac{1}{2}(\phi_{i\max}-\phi_{i\min})\,|\,x_i\,|\,|\,\sigma\,| \tag{10.43}$$

由于

$$\begin{aligned}\bar{\phi}\boldsymbol{x}&=\sum_{i=1}^{n}\bar{\phi}_i x_i,\ \phi\boldsymbol{x}=\sum_{i=1}^{n}\phi_i x_i,\ \phi_{\max}\,|\,\boldsymbol{x}\,|_1\\&=\sum_{i=1}^{n}\phi_{i\max}\,|\,x_i\,|,\ \phi_{\min}\,|\,\boldsymbol{x}\,|_1=\sum_{i=1}^{n}\phi_{i\min}\,|\,x_i\,|\end{aligned} \tag{10.44}$$

则

$$S4=\sum_{i=1}^{n}S4_i \tag{10.45}$$

由此可知

$$S4_i\leqslant 0\text{ 且 }S4\leqslant 0 \tag{10.46}$$

由式(10.32)～式(10.46)，得

$$\sigma\dot{\sigma}\leqslant-\frac{b(\boldsymbol{x},t)}{b_{\min}}k\sigma^2<0 \tag{10.47}$$

即

$$\dot{V}<0 \tag{10.48}$$

10.2.4 仿真实例

被控对象为三阶系统：

$$G(s)=\frac{K}{Js^2(Ts+1)}$$

其中 J 为转动惯量，K 为增益，T 为时间常数。

将被控对象转化为状态方程的形式：

$$\begin{cases}\dot{x}_1=x_2\\ \dot{x}_2=x_3\\ \dot{x}_3=-\dfrac{1}{T}x_3+\dfrac{K}{JT}(u-d_0(\boldsymbol{x},t))\end{cases}$$

其中 $d_0(\boldsymbol{x},t)$ 为外部干扰。

设 J 的变化范围为[100,200]，取 $J=\frac{J_{\max}+J_{\min}}{2}+\frac{J_{\max}-J_{\min}}{2}\sin(2\pi t)$，$J_{\min}=100$，$J_{\max}=200$。

设 T 的变化范围为[0.5,2.0]，取 $T=\frac{T_{\max}+T_{\min}}{2}+\frac{T_{\max}-T_{\min}}{2}\sin(2\pi t)$，$T_{\min}=0.5$，$T_{\max}=2.0$。

假设 $d_0(\boldsymbol{x},t)$ 为一摩擦模型：

$$d_0(\boldsymbol{x},t)=F_c\text{sgn}(x_2)+b_c\cdot x_2$$

其中 x_2 为速度，b_c 为粘度系数，F_c 为库仑摩擦力。

控制输入范围为[-110,110]，取 $K=5000$，则 $d(\boldsymbol{x},t)=-\frac{K}{JT}d_0(\boldsymbol{x},t)$，$\phi_3(\boldsymbol{x},t)=-\frac{1}{T}$，$b(\boldsymbol{x},t)=\frac{K}{JT}$。取 $b_{\min}=12.5$，$b_{\max}=100$，$\phi_1=0$，$\phi_2=0$，$\phi_{3\min}=-2$，$\phi_{3\max}=0$。

在摩擦模型中，取 $F_c=0.5$，$b_c=0.3$，则 $d_{0\min}=-1.5$，$d_{0\max}=1.5$，$d_{\min}=-150$，$d_{\max}=150$。

滑模函数设计为

$$\sigma=c_1(r-\boldsymbol{x}(1))+c_2(\dot{r}-\boldsymbol{x}(2))+(\ddot{r}-\boldsymbol{x}(3))-\sigma_0\mathrm{e}^{-\lambda t}$$

取 $c_1=100$，$c_2=50$，$\lambda=10$。位置指令取正弦信号($S=1$)，$r=A\sin(2\pi Ft)$，$A=0.50$，$F=1.0\text{Hz}$。为了消除抖振，采用如下饱和函数方法：

$$\text{sat}\left(\frac{\sigma}{\varphi}\right)=\begin{cases}1 & \sigma/\varphi>1\\ \dfrac{\sigma}{\varphi} & |\sigma/\varphi|\leqslant 1\\ -1 & \sigma/\varphi<-1\end{cases}$$

在程序中，σ_0，σ_1 和 σ 分别用 s0，s1 和 s 来表示。

$M=1$，$M=2$ 和 $M=3$ 分别为采用切换函数、饱和函数及 PD 控制方法。取 $M=2$，$\varphi=0.15$，采用控制律式(10.22)，仿真结果如图 10-5 和图 10-6 所示。

为了进行比较，采用 PD 控制器。取 $M=3$，将 PD 控制器设计为

$$u(t)=30[r-\boldsymbol{x}(1)]+50[\dot{r}-\boldsymbol{x}(2)]$$

仿真结果如图 10-7 所示。

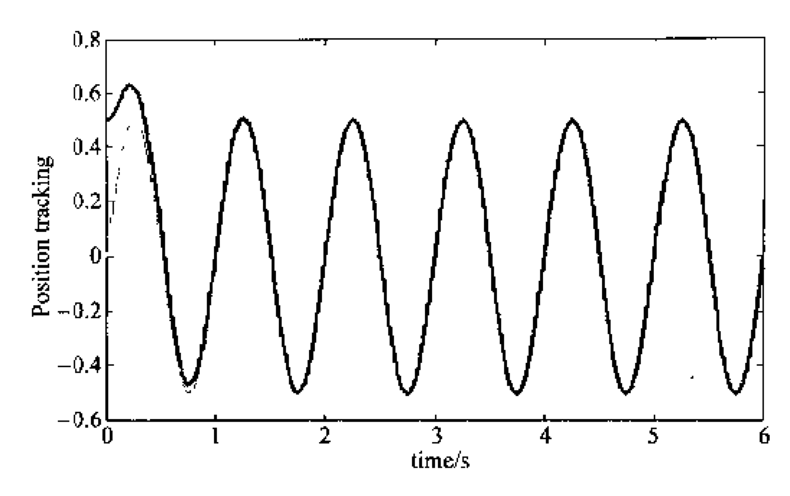


图 10-5　采用饱和函数的位置跟踪

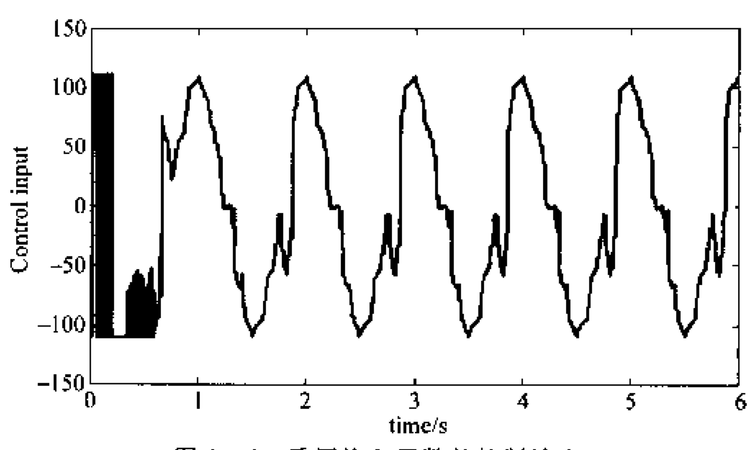


图 10-6　采用饱和函数的控制输入

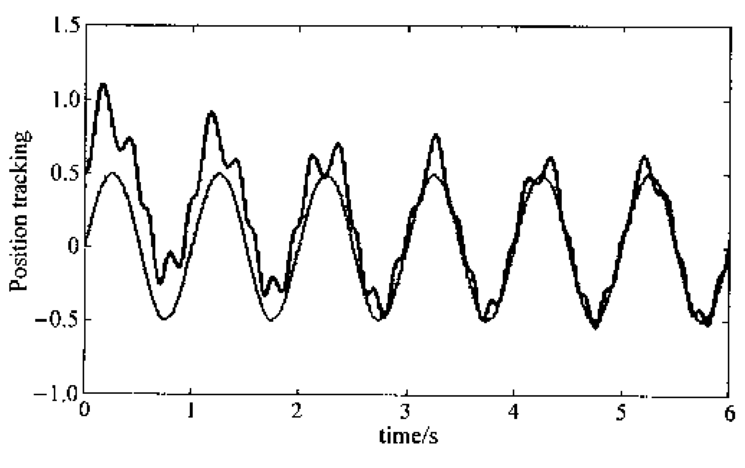


图 10-7　采用 PD 控制的位置跟踪

仿真程序：

(1) Simulink 主程序(如图 10-8 所示)：chap10_2sim.mdl

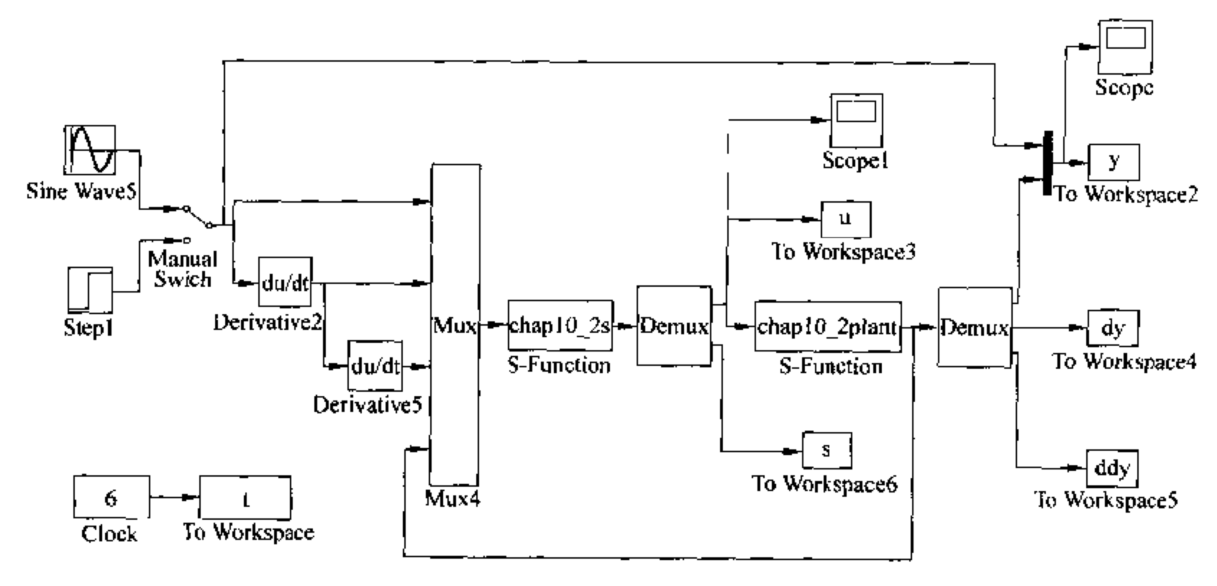


图 10 8 主程序图

(2) 控制器 S 函数程序:chap10_2s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes
global S M x0 fai

sizes = simsizes;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 6;
sizes.DirFeedthrough  = 1;
sizes.NumSampleTimes  = 1; % At least one sample time is needed
sys = simsizes(sizes);
x0=[];
str = [];
ts = [0 0];
function sys=mdlOutputs(t,x,u)

x0 = [0.5;0;0];
```

```
x10 = x0(1);
x20 = x0(2);
x30 = x0(3);
x(1) = u(4);
x(2) = u(5);
x(3) = u(6);

rin = u(1);

S = 1;    %Signal type
if S == 1
    dr = 2 * pi * 0.5 * cos(2 * pi * t);
    ddr = - (2 * pi)^2 * 0.5 * sin(2 * pi * t);
    dddr = - (2 * pi)^3 * 0.5 * cos(2 * pi * t);
    rin0 = 0;
    dr0 = 2 * pi * 0.5;
    ddr0 = 0;
elseif S == 2
    dr = 0;
    ddr = 0;
    dddr = 0;
    rin0 = 0;
    dr0 = 0;
    ddr0 = 0;
end

K = 5000;
%T = 0.05;
T = 0.5 + 1.5 * abs(sin(2 * pi * t));

Jmin = 100;
Jmax = 200;
J = (Jmin + Jmax)/2 + (Jmax - Jmin)/2 * sin(2 * pi * t);

%bmin = K/(Jmax * T);
%bmax = K/(Jmin * T);
bmin = 0.5 * K/Jmax;
bmax = 2 * K/Jmin;
b = (bmin + bmax)/2 + (bmax - bmin)/2 * sin(2 * pi * t);

c1 = 100;c2 = 50;
s1 = c1 * (rin - x(1)) + c2 * (dr - x(2)) + (ddr - x(3));
s0 = c1 * (rin0 - x10) + c2 * (dr0 - x20) + (ddr0 - x30);

s = s1 - s0 * exp( - 10 * t);

fai1 = 0;fai2 = 0;
%fai3 = - 1/T;
fai3_min = - 2;
fai3_max = 0;
```

```
fai3_av = -1;
fai3_cur = 1;

k = 120;

Fc = 0.5;
bc = 0.3;
d = Fc * sign(x(2)) + bc * x(2);
dmin = -150;
dmax = 150;
d_av = 0;

uc = [k * s + dddr - fai3_av * x(3) - d_av + c1 * (dr - x(2)) + c2 * (ddr - x(3))]/bmin;

eq = [(bmax/bmin - 1) * abs(dddr) + fai3_cur * abs(x(3)) + (d_av - dmin) + (bmax/bmin - 1) * [c1 *
abs(dr - x(2)) + c2 * abs(ddr - x(3))]]/bmin;

nmn = 100;
eq = eq + abs(nmn * s0 * exp(-10 * t))/bmin;

M = 3;
if M == 1
    u_vss = eq * sign(s);
    ut - uc + u_vss;
elseif M == 2 % Using saturated function
    fai = 0.15;
    if s/fai>1
        sat = 1;
      elseif abs(s/fai)< = 1
        sat = s/fai;
      elseif s/fai< -1
        sat = -1;
    end
    u_vss = eq * sat;
    ut = uc + u_vss;
elseif M == 3      % PD control
    ut = 30 * (rin - x(1)) + 50 * (dr - x(2));
end

if ut>110
    ut = 110;
end
if ut< -110
    ut = -110;
end
sys(1) = ut;
sys(2) = s;
```

(3) 被控对象 S 函数程序:chap10_2plant.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)
```

```
switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 3;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 3;
sizes.NumInputs       = 1;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 1; % At least one sample time is needed
sys = simsizes(sizes);
x0 = [0.5;0;0];

str = [];
ts = [0 0];
function sys = mdlDerivatives(t,x,u)    % Lugre model
K = 5000;
T = 0.5 + 1.5 * abs(sin(2 * pi * t));
Jmin = 100;
Jmax = 200;
J = (Jmin + Jmax)/2 + (Jmax - Jmin)/2 * sin(2 * pi * t);

Fc = 0.5;
bc = 0.3;
d = Fc * sign(x(2)) + bc * x(2);

sys(1) = x(2);
sys(2) = x(3);
sys(3) = - 1/T * x(3) + K/(J * T) * (u - d);
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
sys(3) = x(3);
```

(4) 作图程序:chap10_2plot.m

```
close all;
figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
```

```
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');

figure(3);
plot(t,s(:,1),'r');
xlabel('time(s)');ylabel('Sliding mode surface');
```

10.3 基于滤波器的机器人滑模控制

10.3.1 机器人动态方程

基于拉格朗日运动学建立的机器人动态方程为

$$M(q)\ddot{q}+C(q,\dot{q})\dot{q}+G(q)=\tau \tag{10.49}$$

其中 q 和 $\dot{q}$ 分别表示机器人各关节的位置和速度，τ 为驱动力矩向量。M,C,G 分别是由机器人的具体结构所决定的 $n\times n,n\times n$ 和 $n\times 1$ 阶函数矩阵，τ 为 $n\times 1$ 阶力矩向量。

对于式(10.49)所示的机器人拉格朗日运动学方程，满足如下四个条件：

(1) 正定对称性：对于任意 q 矩阵，$M(q)$是正定对称的。

(2) 有界性：矩阵函数 $M(q)$和 $C(q,\dot{q})$对于所有 $q,\dot{q}$ 是一致有界的。

(3) 斜对称性：矩阵函数 $\dot{M}(q)-2C(q,\dot{q})$对于任意 $q,\dot{q}$ 是斜对称的。即对任意向量 ξ，有

$$\xi^{\mathrm{T}}(\dot{M}(q)-2C(q,\dot{q}))\xi=0 \tag{10.50}$$

(4) 线性特征：机器人的数学模型对于物理参数是线性的。即如果将矩阵函数 M,C,G 中的定常系数表示为一个向量 θ，则可以定义适当的矩阵 $\phi(q,\dot{q},v,a)$，使得

$$M(q)a+C(q,\dot{q})v+G(q)=\phi(q,\dot{q},v,a)\theta \tag{10.51}$$

成立。其中 v 为速度向量，a 为加速度向量。

在机器人控制器的 Lyapunov 稳定性分析时，通常需要利用上述特性。

10.3.2 滑模控制器的设计

在滑模控制器输出端加入低通滤波器，可将高频抖振控制信号有效地滤除[3]。基于滤波器的机器人滑模控制系统结构如图 10-9 所示。

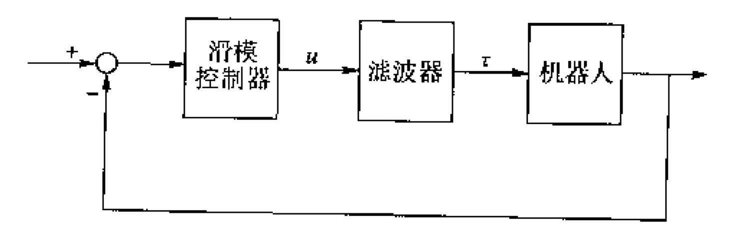


图 10-9 基于滤波器的滑模控制

采用如下低通滤波器：

$$Q(s)=\frac{\lambda_i}{s_i+\lambda_i} \tag{10.52}$$

其中 $\lambda_i>0$。

由图 10-9 可得

$$\dot{\tau}+\Lambda\tau=\Lambda u \tag{10.53}$$

其中 $\Lambda=\mathrm{diag}(\lambda_1,\lambda_2,\cdots,\lambda_n)$，$\lambda_i>0$，$i=1,2,\cdots,n$，$n$ 为控制输入信号的个数。

将式(10.49)代入式(10.53)得

$$M\dddot{q}+\dot{M}\ddot{q}+C\ddot{q}+\dot{C}\dot{q}+\dot{G}+\Lambda M\ddot{q}+\Lambda C\dot{q}+\Lambda G=\Lambda u \tag{10.54}$$

设理想位置指令为 $q_d(t)$，跟踪误差为

$$e(t)=q(t)-q_d(t) \tag{10.55}$$

设计滑模面函数为

$$s(t)=\ddot{e}+\Lambda_1\dot{e}+\Lambda_2 e \tag{10.56}$$

其中 $\Lambda_i=\mathrm{diag}(\lambda_{i1},\lambda_{i2},\cdots,\lambda_{in})$，$\lambda_{ij}>0$，$i=1,2$，$j=1,2,\cdots,n$。

定义 Lyapunov 函数为

$$V=\frac{1}{2}s^{T}Ms \tag{10.57}$$

则

$$\dot{V}=\frac{1}{2}(\dot{s}^{T}Ms+s^{T}\dot{M}s+s^{T}M\dot{s})$$

由于 $M(q)$ 为正定对称阵，则

$$\dot{s}^{T}Ms=s^{T}M\dot{s} \tag{10.58}$$

又根据矩阵函数 $\dot{M}(q)-2C(q,\dot{q})$ 的斜对称性，有

$$s^{T}[\dot{M}(q)-2C(q,\dot{q})]s=0 \tag{10.59}$$

则

$$\begin{aligned}\dot{V}&=s^{T}M\dot{s}+s^{T}Cs=s^{T}(M\dot{s}+Cs)\\&=s^{T}[M(\dddot{q}-\dddot{q}_d+\Lambda_1\ddot{e}+\Lambda_2\dot{e})+C(\ddot{e}+\Lambda_1\dot{e}+\Lambda_2 e)]\end{aligned}$$

由式(10.54)得

$$\begin{aligned}M\dddot{q}&=\Lambda u-\dot{M}\ddot{q}-\Lambda M\ddot{q}-C\ddot{q}-\dot{C}\dot{q}-\Lambda C\dot{q}-\dot{G}-\Lambda G\\&=\Lambda u-(\dot{M}+\Lambda M)\ddot{q}-(\dot{C}+\Lambda C)\dot{q}-(\dot{G}+\Lambda G)-C\ddot{q}\end{aligned}$$

则

$$\begin{aligned}\dot{V}&=s^{T}[\Lambda u-(\dot{M}+\Lambda M)\ddot{q}-(\dot{C}+\Lambda C)\dot{q}-(\dot{G}+\Lambda G)\\&\quad-C\ddot{q}+M(-\dddot{q}_d+\Lambda_1\ddot{e}+\Lambda_2\dot{e})+C(\ddot{e}+\Lambda_1\dot{e}+\Lambda_2 e)]\\&=s^{T}[\Lambda u+M(\Lambda_1\ddot{e}+\Lambda_2\dot{e}-\dddot{q}_d)+C(\Lambda_1\dot{e}+\Lambda_2 e-\ddot{q}_d)\\&\quad-(\dot{M}+\Lambda M)\ddot{q}-(\dot{C}+\Lambda C)\dot{q}-(\dot{G}+\Lambda G)]\\&=s^{T}(\Lambda u+H)\end{aligned}$$

其中

$$\boldsymbol{H}=\boldsymbol{M}(\boldsymbol{\Lambda}_1\ddot{\boldsymbol{e}}+\boldsymbol{\Lambda}_2\dot{\boldsymbol{e}}-\ddot{\boldsymbol{q}}_d)+\boldsymbol{C}(\boldsymbol{\Lambda}_1\dot{\boldsymbol{e}}+\boldsymbol{\Lambda}_2\boldsymbol{e}-\ddot{\boldsymbol{q}}_d)$$
$$-(\dot{\boldsymbol{M}}+\Delta\boldsymbol{M})\ddot{\boldsymbol{q}}-(\dot{\boldsymbol{C}}+\Delta\boldsymbol{C})\dot{\boldsymbol{q}}-(\dot{\boldsymbol{G}}+\Delta\boldsymbol{G}) \tag{10.60}$$

则根据机器人的线性特征式(10.51)，$\boldsymbol{H}$ 可写为

$$\boldsymbol{H}=\boldsymbol{\Gamma}(t,\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}},\boldsymbol{q}_d,\dot{\boldsymbol{q}}_d,\ddot{\boldsymbol{q}}_d,\dddot{\boldsymbol{q}}_d)\boldsymbol{\phi} \tag{10.61}$$

其中 $\boldsymbol{\phi}$ 为未知向量。

存在已知向量 $\boldsymbol{\phi}_0$，使得

$$|\tilde{\phi}_i|=|\phi_i-\phi_{0i}|\leqslant\xi_i \tag{10.62}$$

其中 $\xi_i>0$ 且 $i=1,2,\cdots,n$。

则

$$\dot{V}=\boldsymbol{s}^{\mathrm{T}}(\boldsymbol{\Lambda u}+\boldsymbol{\Gamma\phi})$$

设计滑模控制律为

$$\boldsymbol{u}=-\boldsymbol{\Lambda}^{-1}[\boldsymbol{\Gamma\phi}_0+(\bar{\boldsymbol{\Gamma}}\boldsymbol{\xi})\mathrm{sgn}(\boldsymbol{s})+\boldsymbol{\Psi}\mathrm{sgn}(\boldsymbol{s})] \tag{10.63}$$

其中 $\boldsymbol{\Gamma\phi}_0=\boldsymbol{H}_0$，$\boldsymbol{H}_0$ 为精确对象的 $\boldsymbol{H}$，$\bar{\Gamma}_{ij}=|\Gamma_{ij}|$，$\boldsymbol{\Psi}=\mathrm{diag}(\Psi_1,\Psi_2,\cdots,\Psi_n)$，$\Psi_i>0$，$i=1,2,\cdots,n$，$\mathrm{sgn}(\boldsymbol{s})=[\mathrm{sgn}(s_1)\mathrm{sgn}(s_2)\cdots\mathrm{sgn}(s_n)]$。则

$$\begin{aligned}\dot{V}&=\boldsymbol{s}^{\mathrm{T}}[\boldsymbol{\Gamma}(\boldsymbol{\phi}-\boldsymbol{\phi}_0)-(\bar{\boldsymbol{\Gamma}}\boldsymbol{\xi})\mathrm{sgn}(\boldsymbol{s})-\boldsymbol{\Psi}\mathrm{sgn}(\boldsymbol{s})]\\&=\boldsymbol{s}^{\mathrm{T}}[\boldsymbol{\Gamma}\tilde{\boldsymbol{\phi}}-(\bar{\boldsymbol{\Gamma}}\boldsymbol{\xi})\mathrm{sgn}(\boldsymbol{s})-\boldsymbol{\Psi}\mathrm{sgn}(\boldsymbol{s})]\\&=\sum_{i=1}^{n}\{s_i(\boldsymbol{\Gamma}\tilde{\boldsymbol{\phi}})_i-|s_i|(\bar{\boldsymbol{\Gamma}}\boldsymbol{\xi})_i\}-\boldsymbol{s}^{\mathrm{T}}\boldsymbol{\Psi}\mathrm{sgn}(\boldsymbol{s})\\&\leqslant-\boldsymbol{s}^{\mathrm{T}}\boldsymbol{\Psi}\mathrm{sgn}(\boldsymbol{s})=-\sum_{i=1}^{n}(\Psi_i|s_i|)\leqslant 0\end{aligned}$$

设 $\boldsymbol{\eta}=\bar{\boldsymbol{\Gamma}}\xi+\Psi$，则控制律又可写为

$$\boldsymbol{u}=-\boldsymbol{\Lambda}^{-1}[\boldsymbol{\Gamma\phi}_0+\boldsymbol{\eta}\mathrm{sgn}(\boldsymbol{s})] \tag{10.64}$$

10.3.3　仿真实例之一

以单关节机器人为例，其 SISO 动态方程为

$$M(q)\ddot{q}+C(q,\dot{q})\dot{q}-G(q)+d(t)=\tau \tag{10.65}$$

其中 $M(q)=0.1+0.06\sin(q)$，$C(q,\dot{q})=0.03\cos(q)$，$G(q)=mgl\cos(q)$，$m=0.02$，$g=9.8$，$l=0.05$。$d(t)$表示对象建模的不确定部分，$d(t)=3\sin(2\pi t)$。

可见，式(10.65)满足机器人动态方程的四个条件。

取位置指令为 $r=\sin(2\pi t)$，系统初始状态为$[0.5\quad 0]^{\mathrm{T}}$。本系统控制输入信号为 1 个，$n=1$ 取 $\lambda_{11}=30$，$\lambda_{21}=50$，$\lambda_i=25$，$\eta=50$。控制律式(10.63)写为：$u=-\lambda^{-1}[H_0+\eta\mathrm{sgn}(s)]$，其中 $H_0=M(\lambda_1\ddot{e}+\lambda_2\dot{e}-\ddot{q}_d)+C(\lambda_1\dot{e}+\lambda_2 e-\ddot{q}_d)-(\dot{M}+\lambda M)\ddot{q}-(\dot{C}+\lambda C)\dot{q}-(\dot{G}+\lambda G)$。仿真结果如图 10-10～图 10-12 所示。可见，通过采用低通滤波器，可大大地消除抖振。

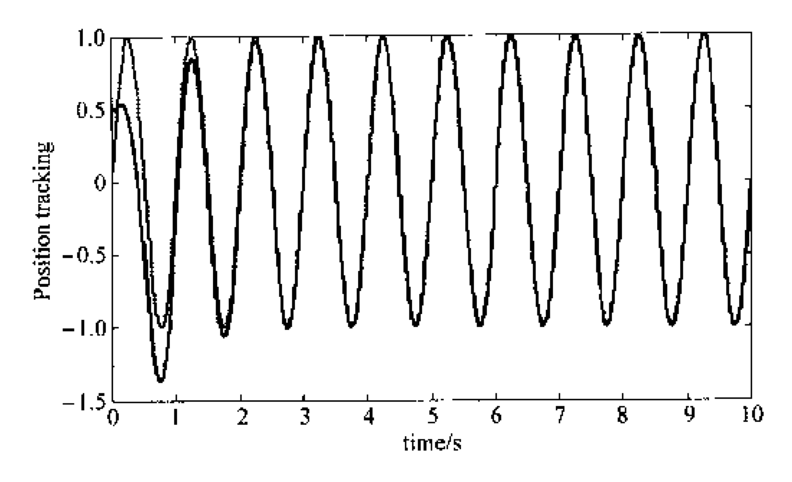


图 10-10　位置跟踪

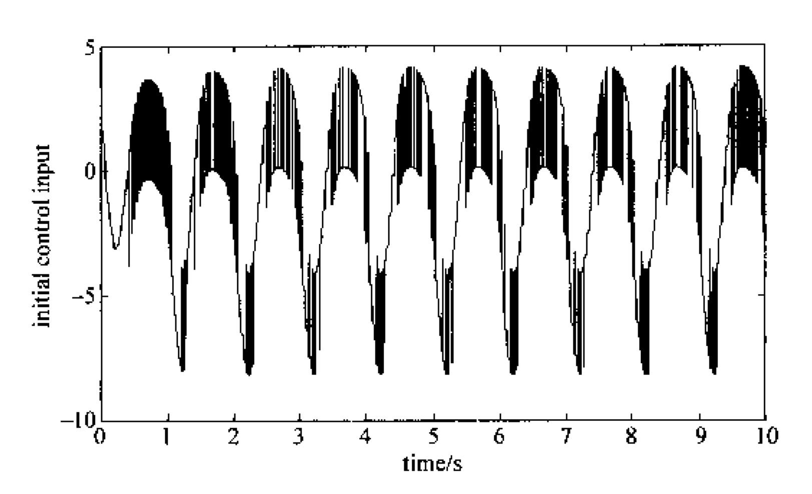


图 10-11　滤波前的控制输入信号 u

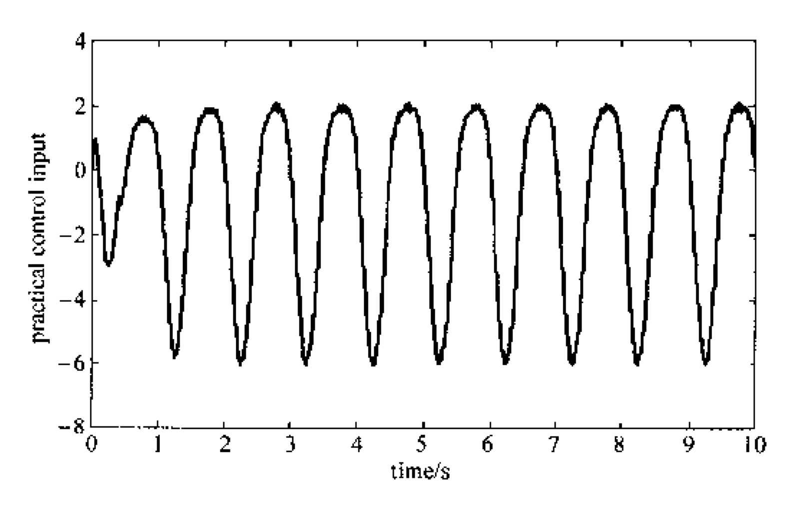


图 10-12　滤波后的控制输入信号 τ

仿真程序：

(1) Simulink 主程序(如图 10-13 所示)：chap10_3sim.mdl

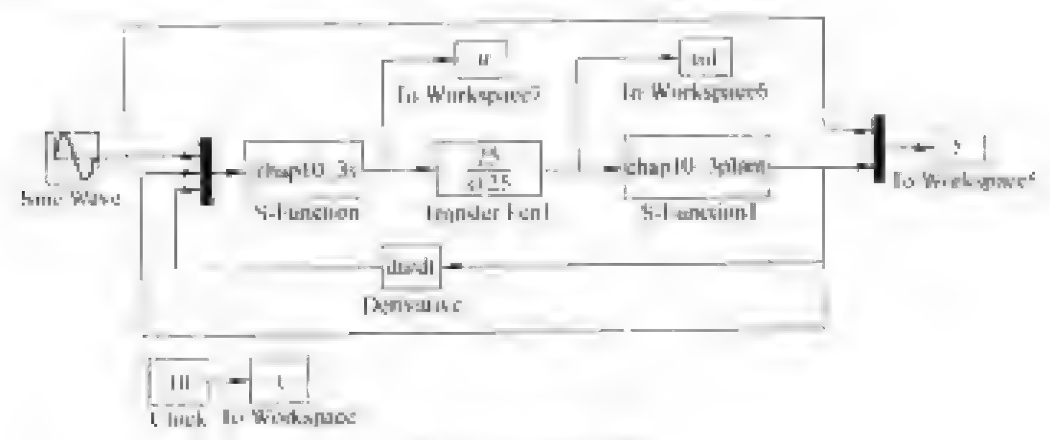


图 10-13 主程序图

(2) 控制器 S 函数程序：chap10_3s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [ ];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 5;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [ ];
str = [ ];
ts  = [0 0];

function sys = mdlOutputs(t,x,u)
thd = u(1);

x1 = u(2);
x2 = u(3);
dx1 = u(4);
dx2 = u(5);
```

```
r = sin(2 * pi * t);
dr = 2 * pi * cos(2 * pi * t);
ddr = - (2 * pi)^2 * sin(2 * pi * t);
dddr = - (2 * pi)^3 * cos(2 * pi * t);
e = x1 - r;
de = x2 - dr;
dde = dx2 - ddr;

n1 = 30;n2 = 50;n = 25;
s = dde + n1 * de + n2 * e;

M = 0.1 + 0.06 * sin(x1);
dM = 0.06 * cos(x1);
C = 0.03 * cos(x1);
dC = - 0.03 * sin(x1);
m = 0.020;
g = 9.8;
l = 0.05;
G = m * g * l * cos(x1);
dG = - m * g * l * sin(x1);

H = M * (n1 * dde + n2 * de - dddr) + C * (n1 * de + n2 * e - ddr) - (dM + n * M) * dx2 - (dC + n * C) * x2
 - (dG + n * G);

xite = 50;
ut = - 1/n * (H + xite * sign(s));

sys(1) = ut;
```

(3) 被控对象 S 函数程序:chap10_3plant.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1; % At least one sample time is needed
sys = simsizes(sizes);
```

```
x0  = [0.5;0];
str = [];
ts  = [0 0];

function sys = mdlDerivatives(t,x,u)     % Time - varying model
tol = u(1);
M = 0.1 + 0.06 * sin(x(1));
C = 0.03 * cos(x(1));
m = 0.020;
g = 9.8;
l = 0.05;
G = m * g * l * cos(x(1));
dt = 3.0 * sin(2 * pi * t);

sys(1) = x(2);
sys(2) = 1/M * ( - C * x(2) - G + tol) + dt;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(4) 作图程序:chap10_3plot. m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('position tracking');

figure(2);
plot(t,u,'r');
xlabel('time(s)');ylabel('initial control input');

figure(3);
plot(t,tol,'r');
xlabel('time(s)');ylabel('practical control input');
```

10.3.4 仿真实例之二

二关节机器人系统(不考虑摩擦力)的 MIMO 动力学模型为

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{d}(t) = \boldsymbol{\tau} \tag{10.66}$$

其中

$$\boldsymbol{M}(\boldsymbol{q}) = \begin{bmatrix} 0.1 + 0.01\cos(q_2) & 0.01\sin(q_2) \\ 0.01\sin(q_2) & 0.1 \end{bmatrix}$$

$$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \begin{bmatrix} -0.005\sin(q_2)\dot{q}_2 & 0.005\cos(q_2)\dot{q}_2 \\ 0.005\cos(q_2)\dot{q}_2 & 0 \end{bmatrix}$$

$$\boldsymbol{G}(\boldsymbol{q}) = \begin{bmatrix} 0.01g\cos(q_1 + q_2) \\ 0.01g\cos(q_1 + q_2) \end{bmatrix}, g = 9.8$$

$$\boldsymbol{d}(t) = [2\sin(2\pi t) \ \ 1.5\cos(2\pi t)]^{\mathrm{T}}$$

二关节的位置指令分别为 $q_{1d} = \cos(\pi t)$ 和 $q_{2d} = \sin(\pi t)$,系统的初始状态为$[q_1 \ q_2 \ q_3 \ q_4] =$

[0.5 0 0.5 0]，本系统控制输入信号为 2 个，$n=2$。取 $\boldsymbol{\Lambda}_1=\begin{bmatrix}\lambda_{11} & 0\\ 0 & \lambda_{12}\end{bmatrix}=\begin{bmatrix}5 & 0\\ 0 & 5\end{bmatrix}$，$\boldsymbol{\Lambda}_2=\begin{bmatrix}\lambda_{21} & 0\\ 0 & \lambda_{22}\end{bmatrix}=\begin{bmatrix}50 & 0\\ 0 & 50\end{bmatrix}$，$\boldsymbol{\Lambda}=\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}=\begin{bmatrix}25 & 0\\ 0 & 25\end{bmatrix}$，控制参数取 $\boldsymbol{\eta}=\begin{bmatrix}1.5 & 0\\ 0 & 1.5\end{bmatrix}$。

采用控制律式(10.63)，$\boldsymbol{u}=-\boldsymbol{A}^{-1}[\boldsymbol{H}_0+\eta\mathrm{sgn}(s)]$，其中 $\boldsymbol{H}=\boldsymbol{M}(\boldsymbol{\Lambda}_1\ddot{\boldsymbol{e}}+\boldsymbol{\Lambda}_2\dot{\boldsymbol{e}}-\ddot{\boldsymbol{q}}_d)+\boldsymbol{C}(\boldsymbol{\Lambda}_1\dot{\boldsymbol{e}}+\boldsymbol{\Lambda}_2\boldsymbol{e}-\ddot{\boldsymbol{q}}_d)-(\dot{\boldsymbol{M}}+\boldsymbol{\Lambda}\boldsymbol{M})\ddot{\boldsymbol{q}}-(\dot{\boldsymbol{C}}+\boldsymbol{\Lambda}\boldsymbol{C})\dot{\boldsymbol{q}}-(\dot{\boldsymbol{G}}+\boldsymbol{\Lambda}\boldsymbol{G})$。仿真结果如图 10-14～图 10-19 所示。

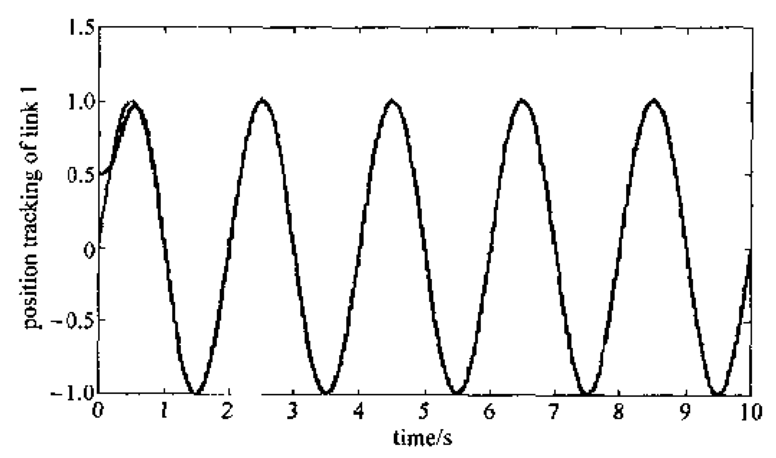


图 10-14　关节 1 的位置跟踪

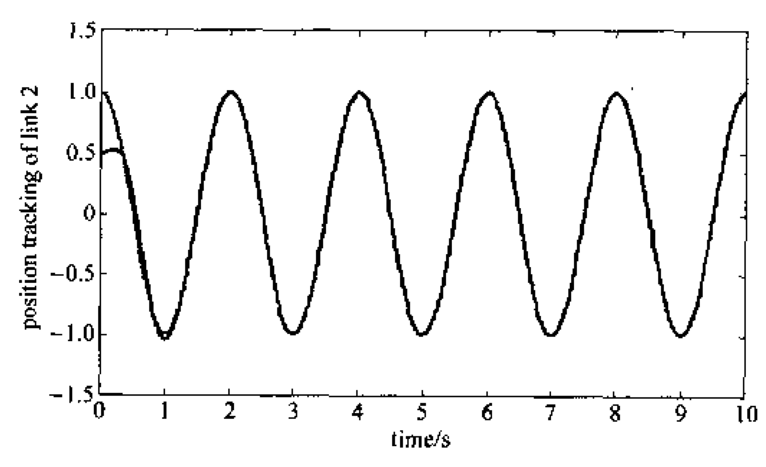


图 10-15　关节 2 的位置跟踪

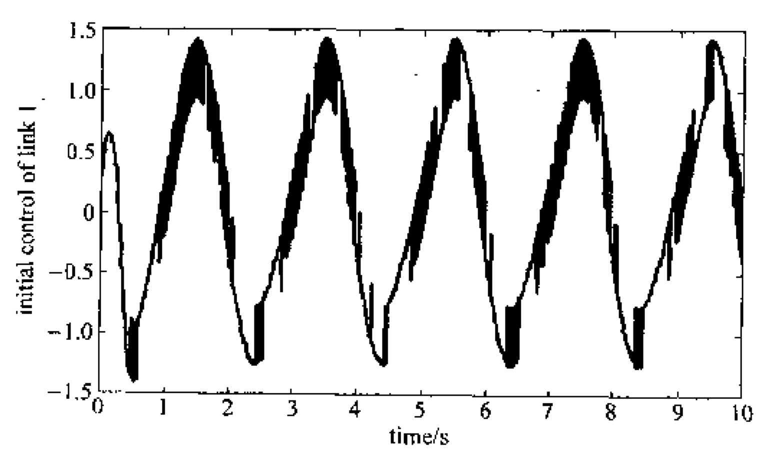


图 10-16　滤波前关节 1 的控制输入

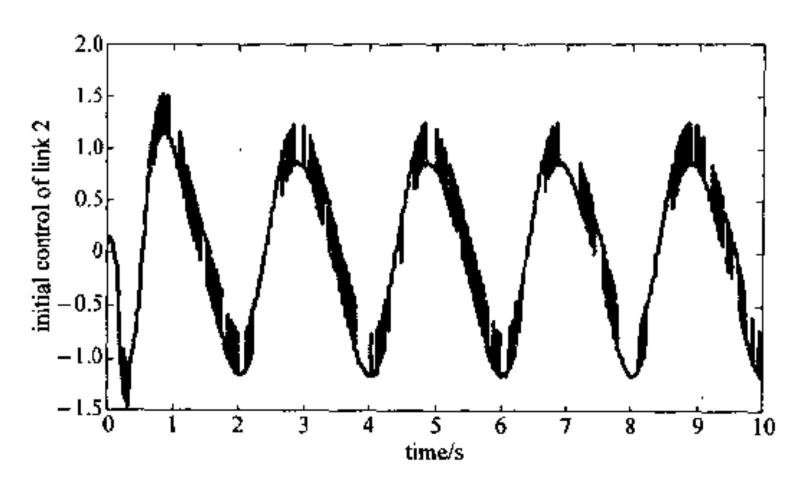


图 10-17　滤波前关节 2 的控制输入

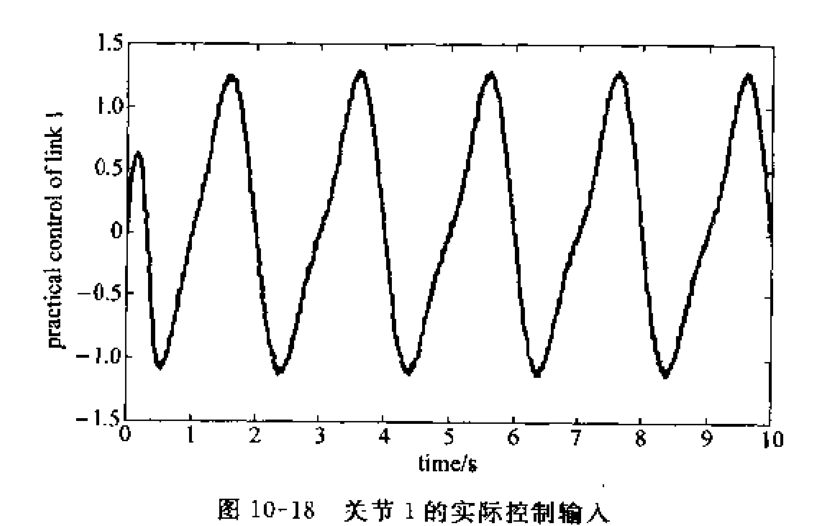


图 10-18　关节 1 的实际控制输入

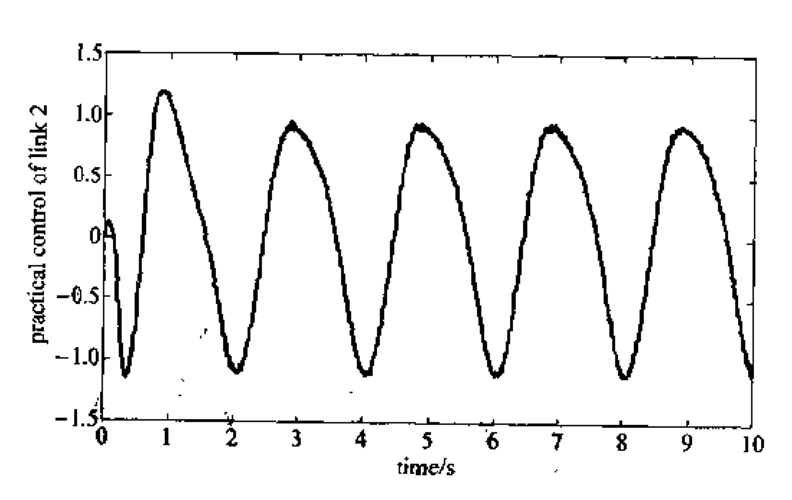


图 10-19　关节 2 的实际控制输入

仿真程序：

(1) Simulink 主程序(如图 10-20 所示)：chap10_4sim.mdl

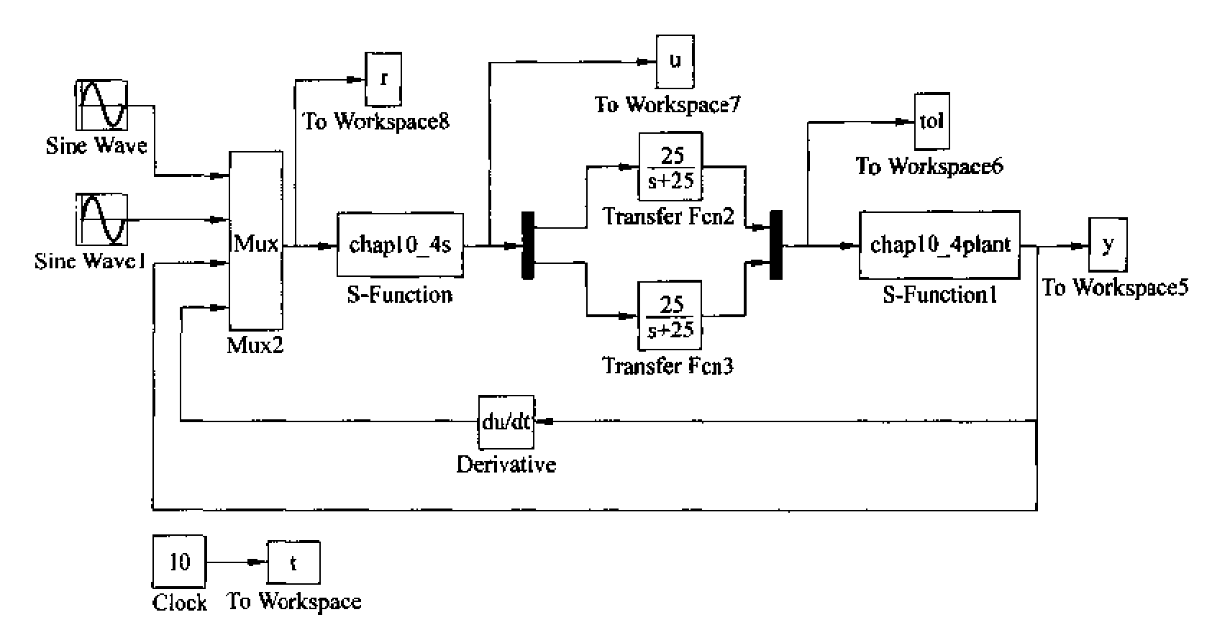


图 10-20　主程序图

(2) 控制器 S 函数程序:chap10_4s.m

```
%S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
%Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
%Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
%Unhandled flags
  case {2, 4, 9 }
    sys  =  [];
%Unexpected flags
  otherwise
    error(['Unhandled flag  =  ',num2str(flag)]);
end

%MdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes  =  simsizes;
sizes.NumContStates   =  0;
sizes.NumDiscStates   =  0;
sizes.NumOutputs      =  2;
sizes.NumInputs       =  10;
sizes.DirFeedthrough  =  1;
sizes.NumSampleTimes  =  0;

sys = simsizes(sizes);
```

```
x0 = [];
str = [];
ts = [];

function sys = mdlOutputs(t,x,u)
q1d = sin(pi * t);
dq1d = pi * cos(pi * t);
ddq1d = - pi^2 * sin(pi * t);
dddq1d = - pi^3 * cos(pi * t);

q2d = cos(pi * t);
dq2d = - pi * sin(pi * t);
ddq2d = - pi^2 * cos(pi * t);
dddq2d = pi^3 * sin(pi * t);

ddqd = [ddq1d;ddq2d];
dddqd = [dddq1d;dddq2d];

x = [u(3);u(4);u(5);u(6)];
dx = [u(7);u(8);u(9);u(10)];

e1 = x(1) - q1d;
de1 = x(2) - dq1d;
dde1 = dx(2) - ddq1d;

e2 = x(3) - q2d;
de2 = x(4) - dq2d;
dde2 = dx(4) - ddq2d;

e = [e1;e2];
de = [de1;de2];
dde = [dde1;dde2];

n1 = [5 0;0 5];
n2 = [50 0;0 50];
n = [25 0;0 25];

s = dde + n1 * de + n2 * e;

g = 9.8;

M11 = 0.1 + 0.01 * cos(x(3));
M12 = 0.01 * sin(x(3));
M21 = M12;
M22 = 0.1;
M = [M11 M12;M21 M22];

dM11 = - 0.01 * sin(x(3)) * x(4);
dM12 = 0.01 * cos(x(3)) * x(4);
dM21 = dM12;
dM22 = 0;
```

```
dM = [dM11 dM12;dM21 dM22];

C11 = - 0.01 * sin(x(3)) * x(4)/2;
C12 = 0.01 * cos(x(3)) * x(4)/2;
C21 = C12;
C22 = 0;
C = [C11 C12;C21 C22];

dC11 = - 0.01 * (cos(x(3)) * x(4) * x(4) + sin(x(3)) * x(4) * dx(4))/2;;
dC12 = - 0.01 * ( - sin(x(3)) * x(4) * x(4) + cos(x(3)) * dx(4))/2;;
dC21 = dC12;
dC22 = 0;
dC = [dC11 dC12;dC21 dC22];

G1 = 0.01 * g * cos(x(1) + x(3));
G2 = 0.01 * g * cos(x(1) + x(3));
G = [G1;G2];
dG1 = - 0.01 * g * sin(x(1) + x(3)) * (x(2) + x(4));
dG2 = - 0.01 * g * sin(x(1) + x(3)) * (x(2) + x(4));
dG = [dG1;dG2];

H0 = M * (n1 * dde + n2 * de - dddqd) + C * (n1 * de + n2 * e - ddqd) - (dM + n * M) * [dx(2);dx(4)] -
(dC + n * C) * [x(2);x(4)] - (dG + n * G);

xite = [6 0;0 5];
ut = - inv(n) * (H0 + xite * sign(s));

sys(1) = ut(1);
sys(2) = ut(2);
```

(3) 被控对象 S 函数程序:chap10_4plant. m

```
% S - function for continuous state equation
function [sys,x0,str,ts] = s_function(t,x,u,flag)

switch flag,
% Initialization
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
% Outputs
  case 3,
    sys = mdlOutputs(t,x,u);
% Unhandled flags
  case {2, 4, 9 }
    sys = [];
% Unexpected flags
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
```

```
%mdlInitializeSizes
function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 4;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 4;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 0;

sys = simsizes(sizes);
x0 = [0.5,0,0.5,0];
str = [];
ts = [];

function sys = mdlDerivatives(t,x,u)
tol = [u(1);u(2)];

g = 9.8;

M11 = 0.1 + 0.01 * cos(x(3));
M12 = 0.01 * sin(x(3));
M21 = M12;
M22 = 0.1;
M = [M11 M12;M21 M22];

C11 = - 0.01 * sin(x(3)) * x(4)/2;
C12 = 0.01 * cos(x(3)) * x(4)/2;
C21 = C12;
C22 = 0;
C = [C11 C12;C21 C22];

G1 = 0.01 * g * cos(x(1) + x(3));
G2 = 0.01 * g * cos(x(1) + x(3));
G = [G1;G2];

dt = [2 * sin(2 * pi * t);1.5 * cos(2 * pi * t)];

S = inv(M) * (tol - C * [x(2);x(4)] - G) + dt;

sys(1) = x(2);
sys(2) = S(1);
sys(3) = x(4);
sys(4) = S(2);

function sys = mdlOutputs(t,x,u)

sys(1) = x(1);
sys(2) = x(2);
sys(3) = x(3);
sys(4) = x(4);
```

(4) 作图程序:chap10_4plot.m

```
close all;

figure(1);
plot(t,r(:,1),'r',t,y(:,1),'b');
xlabel('time(s)');ylabel('position tracking of link 1');
figure(2);
plot(t,r(:,2),'r',t,y(:,3),'b');
xlabel('time(s)');ylabel('position tracking of link 2');

figure(3);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('initial control of link 1');
figure(4);
plot(t,u(:,2),'r');
xlabel('time(s)');ylabel('initial control of link 2');

figure(5);
plot(t,tol(:,1),'r');
xlabel('time(s)');ylabel('practical control of link 1');
figure(6);
plot(t,tol(:,2),'r');
xlabel('time(s)');ylabel('practical control of link 2');
```

10.4　一种基于积分切换函数的自适应滑模控制

10.4.1　系统描述

考虑 SISO 不确定系统:

$$\dot{\boldsymbol{X}}(t)=(\boldsymbol{A}+\Delta\boldsymbol{A})\boldsymbol{X}(t)+(\boldsymbol{B}+\Delta\boldsymbol{B})U(t)+(D+\Delta D)T_L \tag{10.67}$$

其中 $\Delta\boldsymbol{A}$,$\Delta\boldsymbol{B}$ 和 ΔD 为系统参数和外加干扰的不确定性,$\boldsymbol{B}>\boldsymbol{0}$。

$$\dot{\boldsymbol{X}}(t)=\boldsymbol{A}\boldsymbol{X}(t)+\boldsymbol{B}[U(t)+E(t)] \tag{10.68}$$

式中

$$E(t)=\boldsymbol{B}^{+}\Delta\boldsymbol{A}\boldsymbol{X}(t)+\boldsymbol{B}^{+}\Delta\boldsymbol{B}U(t)-\boldsymbol{B}^{+}(D+\Delta D)T_L \tag{10.69}$$

其中 $\boldsymbol{B}^{+}=(\boldsymbol{B}^{T}\boldsymbol{B})^{-1}\boldsymbol{B}^{T}$。

10.4.2　积分型切换函数的设计

将积分型切换函数设计为

$$s(t)=\boldsymbol{C}\left[\boldsymbol{X}(t)-\int_0^t(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{K})\boldsymbol{X}(t)\mathrm{d}t\right] \tag{10.70}$$

其中 $\boldsymbol{C}$ 为正常数构成的矩阵,$\boldsymbol{K}$ 为状态反馈增益矩阵。

当系统状态处于滑模面上时,$s(t)=\dot{s}(t)=0$,即

$$\dot{\boldsymbol{X}}(t) = (\boldsymbol{A}+\boldsymbol{BK})\boldsymbol{X}(t) \tag{10.71}$$

此时，通过设计状态反馈增益矩阵 $\boldsymbol{K}$，可达到理想的控制效果。

10.4.3 滑模控制器的设计

滑模控制器设计为

$$U(t) = \boldsymbol{B}^{-1}(\dot{\boldsymbol{R}}-\boldsymbol{AR})-\boldsymbol{KX}_e(t)+f\text{sgn}(s(t)) \tag{10.72}$$

其中 $f \geqslant |E(t)|$，$\boldsymbol{X}_e(t)=\boldsymbol{R}-\boldsymbol{X}=\begin{bmatrix} r \\ \dot{r} \end{bmatrix}-\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$，$\boldsymbol{K}<0$，$s(t)=\boldsymbol{C}[\boldsymbol{X}_e(t)-\int_0^t(\boldsymbol{A}+\boldsymbol{BK})\boldsymbol{X}_e(t)\mathrm{d}t]$。

稳定性分析：

$$\begin{aligned}
\dot{s} &= \boldsymbol{C}\dot{\boldsymbol{X}}_e - \boldsymbol{C}(\boldsymbol{A}+\boldsymbol{BK})\boldsymbol{X}_e = \boldsymbol{C}(\dot{\boldsymbol{R}}-\dot{\boldsymbol{X}})-\boldsymbol{C}(\boldsymbol{A}+\boldsymbol{BK})\boldsymbol{X}_e \\
&= \boldsymbol{C}\dot{\boldsymbol{R}}-\boldsymbol{C}(\boldsymbol{AX}+\boldsymbol{B}U+\boldsymbol{B}E)-\boldsymbol{CAX}_e-\boldsymbol{CBKX}_e \\
&= \boldsymbol{C}(\dot{\boldsymbol{R}}-\boldsymbol{AX}-\dot{\boldsymbol{R}}+\boldsymbol{AR}+\boldsymbol{BKX}_e-\boldsymbol{B}f\text{sgn}(s)+\boldsymbol{B}E)-\boldsymbol{CAX}_e-\boldsymbol{CBKX}_e \\
&= \boldsymbol{C}(-\boldsymbol{B}f\text{sgn}(s)-\boldsymbol{B}E) = \boldsymbol{CB}(E-f\text{sgn}(s))
\end{aligned}$$

则

$$s\dot{s} = \boldsymbol{CB}(sE-f|s|) \leqslant 0$$

10.4.4 仿真实例

被控对象为

$$\dot{\boldsymbol{X}}(t) = \boldsymbol{AX}(t)+\boldsymbol{B}[U(t)+E(t)]$$

其中 $\boldsymbol{A}=\begin{bmatrix} 0 & 1 \\ 0 & -25 \end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix} 0 \\ 133 \end{bmatrix}$，$E=0.1\sin(2\pi t)$。

系统初始状态为 $[0.5 \quad 0]$，取 $\boldsymbol{C}=[3 \quad 1]$，$\boldsymbol{K}=[-100 \quad -10]$，$f=0.10$，位置指令取 $r=0.5\sin(2\pi t)$，采用控制律式(10.72)，仿真结果如图 10-21 和图 10-22 所示。

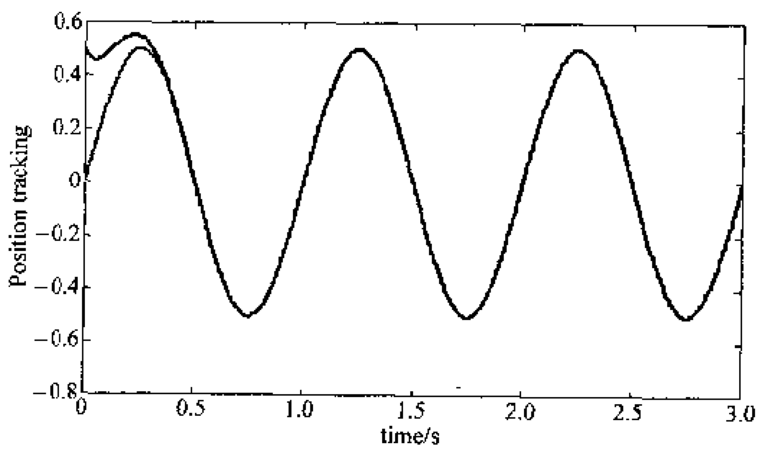


图 10-21 位置跟踪

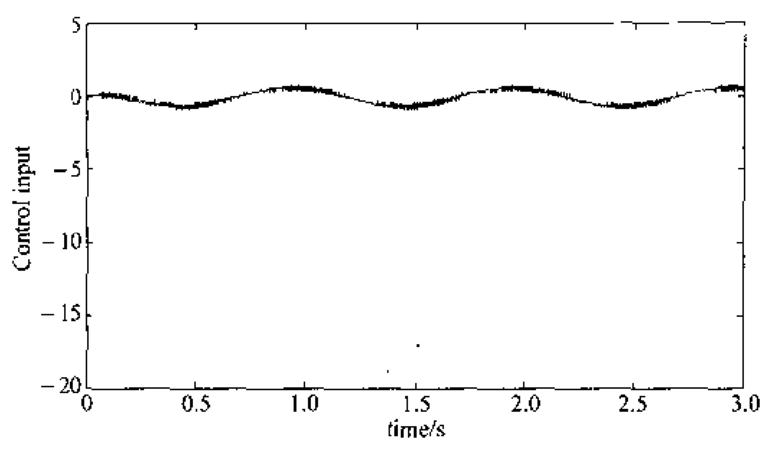


图 10-22　控制输入信号

仿真程序：

(1) Simulink 主程序(如图 10-23 所示)：chap10_5sim. mdl

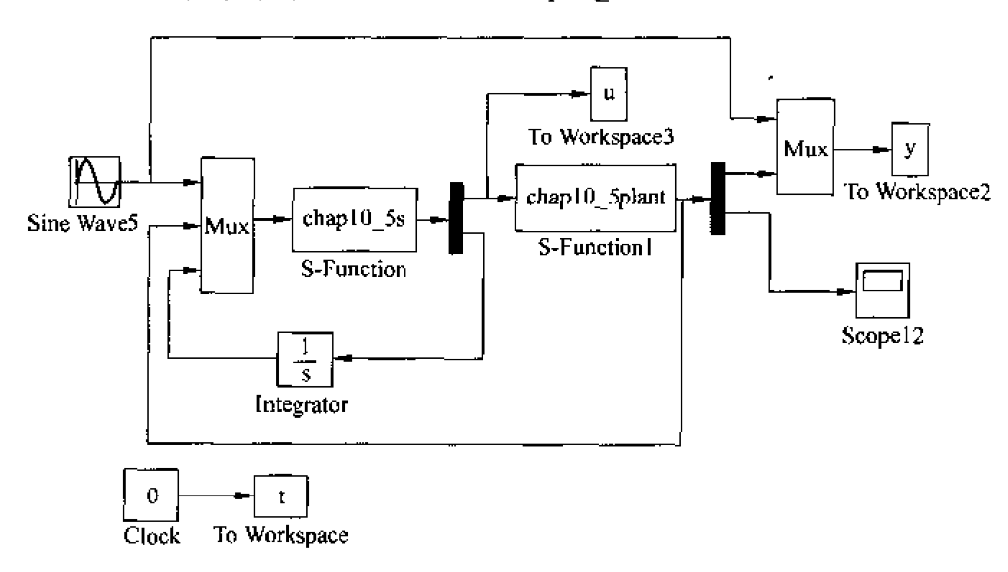


图 10-23　主程序图

(2) 控制器 S 函数程序：chap10_5s. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes
```

```
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1; % At least one sample time is needed
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0 0];

function sys = mdlOutputs(t,x,u)
r = u(1);
dr = 0.5 * 2 * pi * cos(2 * pi * t);
ddr = -0.5 * (2 * pi)^2 * sin(2 * pi * t);

R = [r;dr];
dR = [dr;ddr];
X = [u(2);u(3)];

xe = R - X;

A = [0 1;0 -25];
B = [0;133];

K = [-100,-10];
c1 = 3;
C = [c1,1];

int = C * (A + B * K) * xe;
s = C * (xe - u(4));

f = 0.10;
ut = inv(B' * B) * B' * (dR - A * R) - K * xe + f * sign(s);

sys(1) = ut;
sys(2) = int;
```

(3) 被控对象 S 函数程序:chap10_5plant.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
```

```
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [0.5;0];
str = [];
ts = [0 0];

function sys = mdlDerivatives(t,x,u)
E = 0.1 * sin(2 * pi * t);

A = [0 1;0 -25];
B = [0;133];

ut = u(1);
sys(1) = x(2);
sys(2) = A(2,2) * x(2) + B(2) * (ut + E);

function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(4) 作图程序:chap10_5plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('position tracking');

figure(2);
plot(t,u,'r');
xlabel('time(s)');ylabel('control input');
```

10.4.5 自适应滑模控制器的设计

在实际控制中,总不确定部分 E 的上界很难得到。为此,需要采用自适应控制方法,实现对 E 的上界的自适应估计。

针对系统式(10.68),假设 $\bar{r}$ 为 $E(t)$的上界,$|E(t)|<r$,$\hat{r}$ 为 $E(t)$上界的估计值。滑模控制律设计为

$$U(t) = \boldsymbol{B}^{-1}(\dot{\boldsymbol{R}} - \boldsymbol{AR}) - \boldsymbol{KX}_e(t) + \hat{r}\mathrm{sgn}(s(t)) \tag{10.73}$$

设计自适应律为

$$\dot{\hat{r}}(t)=\frac{1}{\alpha}\mid s(t)\boldsymbol{CB}\mid \tag{10.74}$$

其中 α 为自适应项的增益，$\alpha>0$。

稳定性分析：

定义 Lyapunov 函数

$$V=\frac{1}{2}s^2+\frac{1}{2}\alpha\tilde{r}(t)^2$$

其中 $\tilde{r}(t)=\hat{r}(t)-r$。则

$$\dot{V}=s\dot{s}+\alpha\tilde{r}\dot{\hat{r}}(t)$$

$$\begin{aligned}\dot{s}&=\boldsymbol{C}\dot{\boldsymbol{X}}_e-\boldsymbol{C}(\boldsymbol{A}+\boldsymbol{BK})\boldsymbol{X}_e=\boldsymbol{C}(\dot{\boldsymbol{R}}-\dot{\boldsymbol{X}})-\boldsymbol{C}(\boldsymbol{A}+\boldsymbol{BK})\boldsymbol{X}_e\\&=\boldsymbol{C}\dot{\boldsymbol{R}}-\boldsymbol{C}(\boldsymbol{AX}+\boldsymbol{B}U+\boldsymbol{B}E)-\boldsymbol{CAX}_e-\boldsymbol{CBKX}_e\\&=\boldsymbol{C}(\dot{\boldsymbol{R}}-\boldsymbol{AX}-\dot{\boldsymbol{R}}+\boldsymbol{AR}+\boldsymbol{BKX}_e-\boldsymbol{B}\hat{r}\operatorname{sgn}(s)+\boldsymbol{B}E)-\boldsymbol{CAX}_e-\boldsymbol{CBKX}_e\\&=\boldsymbol{C}(-\boldsymbol{B}\hat{r}\operatorname{sgn}(s)+\boldsymbol{B}E)=\boldsymbol{CB}(E-\hat{r}\operatorname{sgn}(s))\end{aligned}$$

$$\alpha\tilde{r}\dot{\hat{r}}=\alpha\tilde{r}\frac{1}{\alpha}\mid s\boldsymbol{CB}\mid=\tilde{r}\mid s\boldsymbol{CB}\mid=\boldsymbol{CB}(\hat{r}-r)\mid s\mid$$

则

$$\dot{V}=\boldsymbol{CB}(sE-\hat{r}\mid s\mid)+\boldsymbol{CB}(\hat{r}-r)\mid s\mid=\boldsymbol{CB}(sE-r\mid s\mid)\leqslant 0$$

10.4.6 仿真实例

被控对象为

$$\dot{\boldsymbol{X}}(t)=\boldsymbol{AX}(t)+\boldsymbol{B}[U(t)+E(t)]$$

其中 $\boldsymbol{A}=\begin{bmatrix}0&1\\0&-25\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}0\\133\end{bmatrix}$，$E=0.1\sin(2\pi t)$。

系统初始状态为$[0.5\quad 0]^{\mathrm{T}}$，取 $\boldsymbol{C}=[3\quad 1]$，$\boldsymbol{K}=[-100\quad -10]$，$\alpha=250$，位置指令取 $r=0.5\sin(2\pi t)$，采用控制律式(10.73)及自适应律式(10.74)，仿真结果如图 10-24～图 10-26所示。

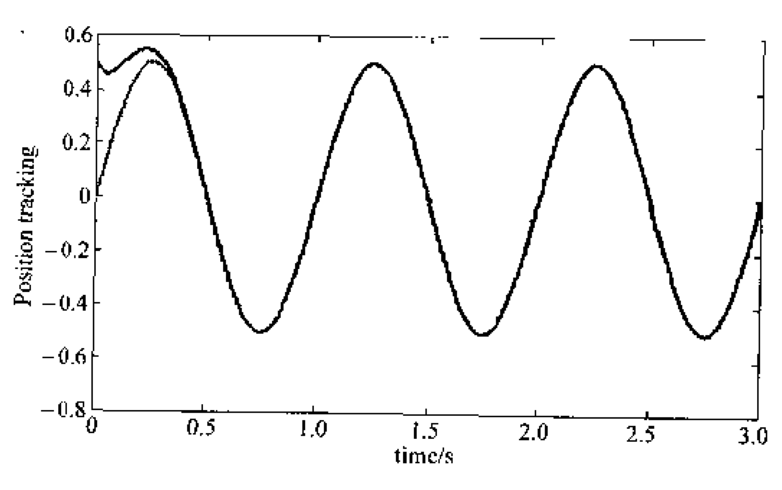


图 10-24 位置跟踪

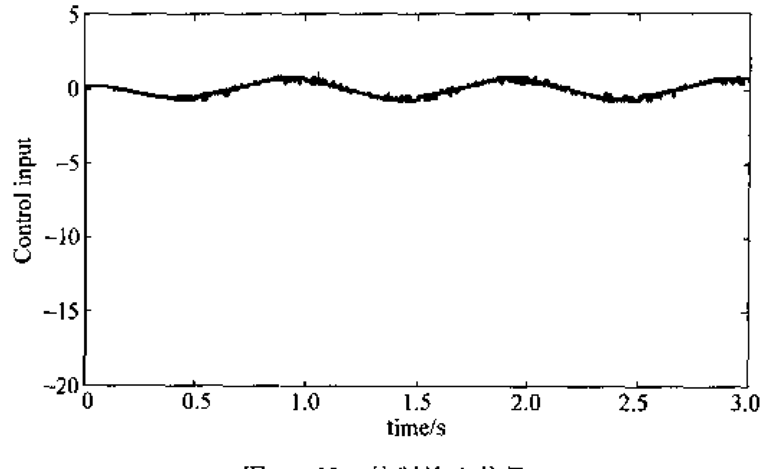


图 10-25　控制输入信号

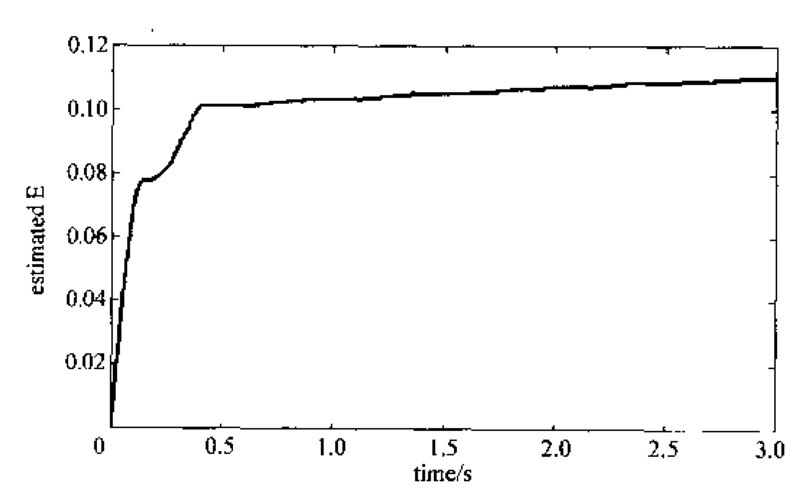


图 10-26　自适应项 $\hat{r}(t)$的变化

仿真程序：

(1) Simulink 主程序(如图 10-27 所示)：chap10_6sim. mdl

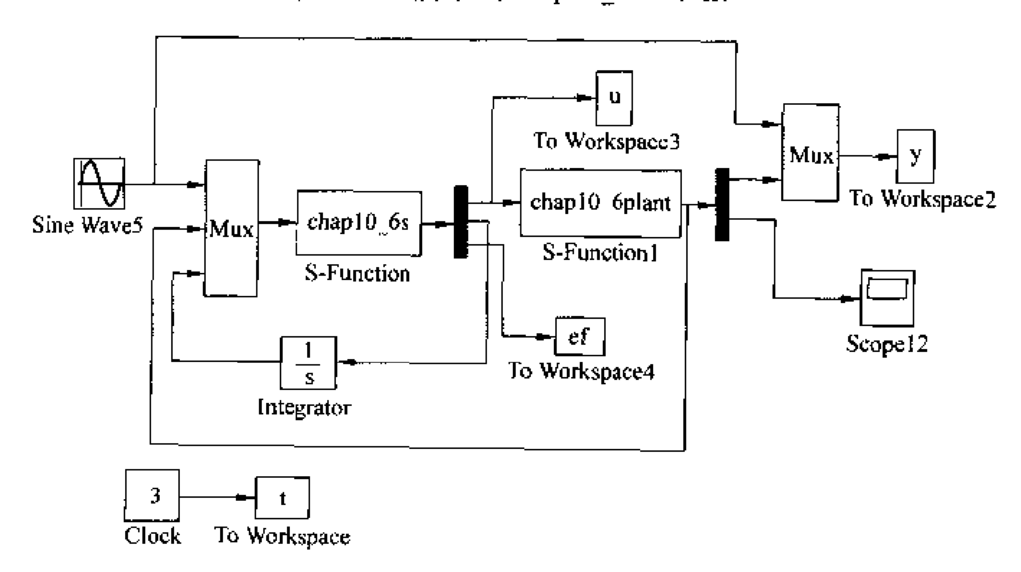


图 10-27　主程序图

（2）控制器 S 函数程序：chap10_6s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 3;
sizes.NumInputs      = 4;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [0];
str = [];
ts  = [0 0];

function sys = mdlDerivatives(t,x,u)
alfa = 250;
r = u(1);
dr = 0.5 * 2 * pi * cos(2 * pi * t);

R = [r;dr];
X = [u(2);u(3)];

xe = R - X;

C = [3,1];
s = C * xe - u(4);

B = [0;133];
sys(1) = 1/alfa * abs(s * C * B);
function sys = mdlOutputs(t,x,u)
r = u(1);
dr = 0.5 * 2 * pi * cos(2 * pi * t);
ddr = -0.5 * (2 * pi)^2 * sin(2 * pi * t);

R = [r;dr];
```

```
dR = [dr;ddr];
X = [u(2);u(3)];

xe = R - X;

A = [0 1;0 -25];
B = [0;133];

K = [-100,-10];
C = [3,1];

int = C * (A + B * K) * xe;
s = C * xe - u(4);

ut = inv(B' * B) * B' * (dR - A * R) - K * xe + x(1) * sign(s);

sys(1) = ut;
sys(2) = int;
sys(3) = x(1);
```

(3) 被控对象 S 函数程序：chap10_6plant.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [0.5;0];
str = [];
ts = [0 0];

function sys = mdlDerivatives(t,x,u)
```

```
E = 0.1 * sin(2 * pi * t);

A = [0 1;0 -25];
B = [0;133];

ut = u(1);
sys(1) = x(2);
sys(2) = A(2,2) * x(2) + B(2) * (ut + E);

function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(4) 作图程序:chap10_6plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,2),'b');
xlabel('time(s)');ylabel('position tracking');

figure(2);
plot(t,u,'r');
xlabel('time(s)');ylabel('control input');

figure(3);
plot(t,ef,'r');
xlabel('time(s)');ylabel('estimated E');
```

10.5 基于积分型切换增益的滑模控制

10.5.1 系统描述

考虑单关节机器人动态方程:

$$I\ddot{x} + D\dot{x} = Gu(t) \tag{10.75}$$

其中 I 为机器人的转动惯量,D 为机器人的粘性摩擦系数,G 为力矩增益。

将式(10.75)写为状态方程的形式:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}u(t) \tag{10.76}$$

其中 $\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{D}{I} \end{bmatrix}$,$\boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{G}{I} \end{bmatrix}$。

10.5.2 常规滑模控制器的设计

切换函数设计为

$$\sigma(t) = Cx \tag{10.77}$$

其中 C 满足滑模稳定条件，且 $CB>0$。

定义 Lyapunov 函数为

$$V = \frac{1}{2}\sigma^2 \tag{10.78}$$

则

$$\dot{V} = \sigma\dot{\sigma} = \sigma C(Ax + Bu) \tag{10.79}$$

要使 $\dot{V}<0$，需要

$$\begin{cases} u < -(CB)^{-1}CAx & \text{if} \quad \sigma > 0 \\ u > -(CB)^{-1}CAx & \text{if} \quad \sigma < 0 \\ u = 0 & \text{if} \quad \sigma = 0 \end{cases} \tag{10.80}$$

基于上述原理，可设计以下两种常规滑模控制器。

1. 切换滑模控制

控制律设计为

$$u = -K\mathrm{sgn}(\sigma) \tag{10.81}$$

其中 $K>|(CB)^{-1}CAx|$。

稳定性分析：

将控制律式(10.81)代入式(10.79)，得

$$\begin{aligned}\dot{V} &= \sigma CAx + \sigma CBu = \sigma CAx - \sigma CBK\mathrm{sgn}(\sigma) \\ &= \sigma CAx - CBK\,|\sigma| < \sigma CAx - |\sigma CAx| < 0 \quad (\text{当 } \sigma \neq 0 \text{ 时})\end{aligned}$$

2. 等效滑模控制

当 $\dot{\sigma}(t)=0$ 时，由式(10.76)和式(10.77)得

$$u_{eq} = -(CB)^{-1}CAx$$

控制律设计为

$$u = u_{eq} + u_d \tag{10.82}$$

其中

$$u_d = -K_e\mathrm{sgn}(\sigma),\, K_e > 0$$

稳定性分析：

将控制律式(10.82)代入式(10.79)，得：$\dot{V}=-K_eCB|\sigma|<0$（当 $\sigma\neq0$ 时）。

在控制律式(10.81)和式(10.82)中，为了保证鲁棒性，控制器的增益必须足够大，这就不可避免地产生抖振。

10.5.3　具有积分型切换增益的滑模控制器

由于在滑动模态阶段，当 σ 趋近于零时，σ 的积分趋近于零。为此，文献[5]提出了具有积分形式增益的滑模控制器，即在增益项中包含切换函数 σ 积分的绝对值，使当 σ 趋近于零时，切换项的增益趋近于零，从而消除抖振。

1. 简单的积分型增益滑模控制器

控制律设计为

$$u = u_{eq} + u_d \tag{10.83}$$

其中

$$u_{eq} = -(\boldsymbol{CB})^{-1}\boldsymbol{CAx}$$

$$u_d = -K_i\left|\int_0^t \sigma \mathrm{d}t\right|\mathrm{sgn}(\sigma), K_i > 0$$

稳定性分析：

将控制律式(10.83)代入式(10.79)，得：$\dot{V} = -K_i\boldsymbol{CB}\left|\int_0^t \sigma \mathrm{d}t\right||\sigma| < 0$(当 $\sigma \neq 0$ 时)。

当不在滑动模态阶段时，σ 值较大，σ 的积分也较大，造成切换项的增益较大，抖振严重。为此，文献[5]又提出了一种新型的具有积分形式增益的滑模控制器，即在积分项中引入负的权值 K_f，有效地避免了当系统不在滑动模态阶段时切换项增益的增大。

2. 加权积分型增益的滑模控制器

控制律设计为

$$u = u_{eq} + u_d \tag{10.84}$$

其中

$$u_{eq} = -(\boldsymbol{CB})^{-1}\boldsymbol{CAx}$$

$$u_d = -K_w|\rho|\mathrm{sgn}(\sigma), K_w > 0$$

$$\rho = \int_0^t (K_f\rho + \sigma)\mathrm{d}t, K_f < 0$$

在积分项 ρ 的表达式中，当 $\rho > 0$ 时，$K_f\rho < 0$，当 $\rho < 0$ 时，$K_f\rho > 0$。有效地避免了当系统不在滑动模态阶段时切换项增益的增大。

稳定性分析：

将控制律式(10.84)代入式(10.79)，得：$\dot{V} = -K_w\boldsymbol{CB}|\rho||\sigma| < 0$(当 $\sigma \neq 0$ 时)。

10.5.4 仿真实例

被控对象为

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{Ax}(t) + \boldsymbol{B}u(t)$$

其中 $\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{D}{I} \end{bmatrix}$，$\boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{G}{I} \end{bmatrix}$。

系统初始状态为$[0.5 \quad 0]^{\mathrm{T}}$。在切换函数中，取 $\boldsymbol{C} = [5 \quad 1]$。当 $M=1$ 时，采用控制律式(10.81)，$K = |(\boldsymbol{CB})^{-1}\boldsymbol{CAx}| + 1.0$，仿真结果如图 10-28～图 10-30 所示。当 $M=2$ 时，采用控制律式(10.82)，$K_c = 1.0$，仿真结果如图 10-31～图 10-33 所示。当 $M=3$ 时，采用控制律式(10.83)，$K_i = 1.0$，仿真结果如图 10-34～图 10-36 所示。当 $M=4$ 时，采用控制律式(10.84)，$K_w = 5.0$，$K_f = -5.0$，仿真结果如图 10-37～图 10-39 所示。

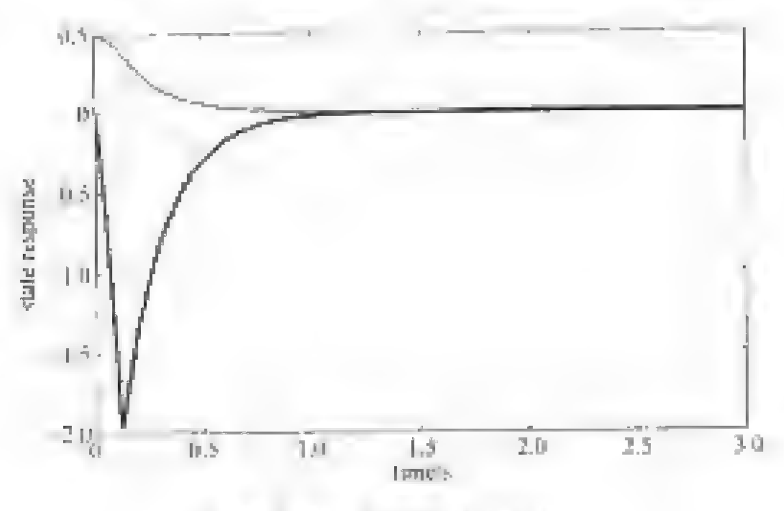


图10-28 状态响应(M=1)

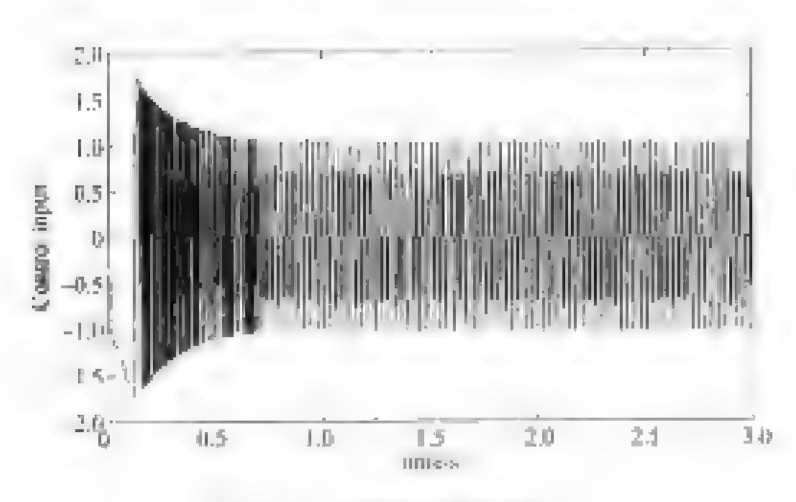


图10-29 控制输入信号(M=1)

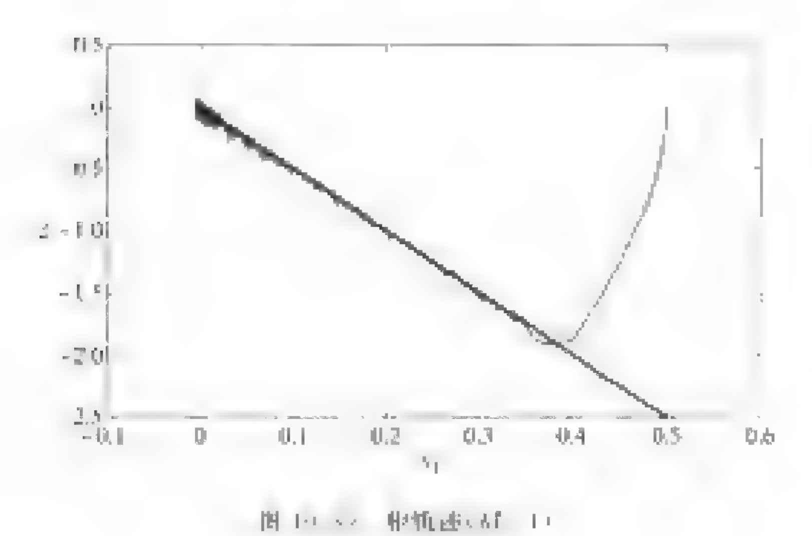


图10-30 相轨迹(M=1)

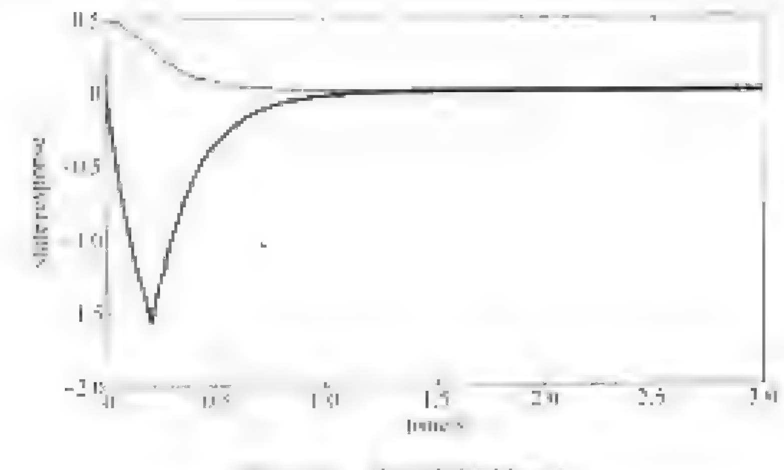


图 10-31　状态响应(M=2)

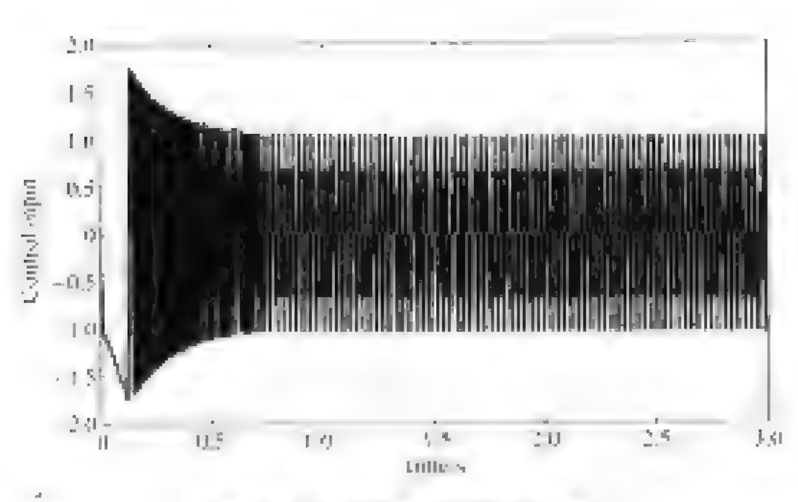


图 10-32　控制输入信号(M=2)

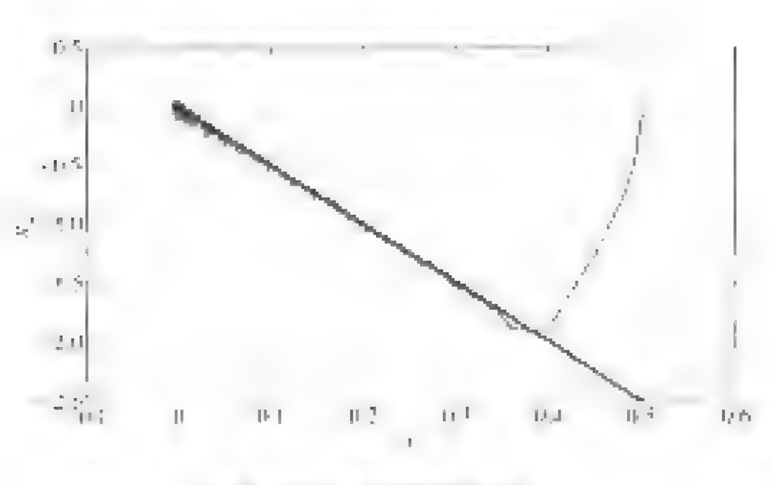


图 10-33　相轨迹(M=2)

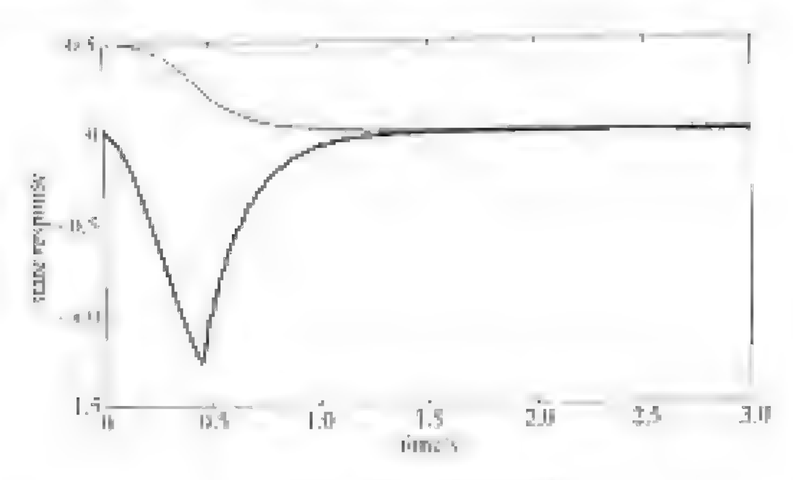


图10-34 状态响应(M=1)

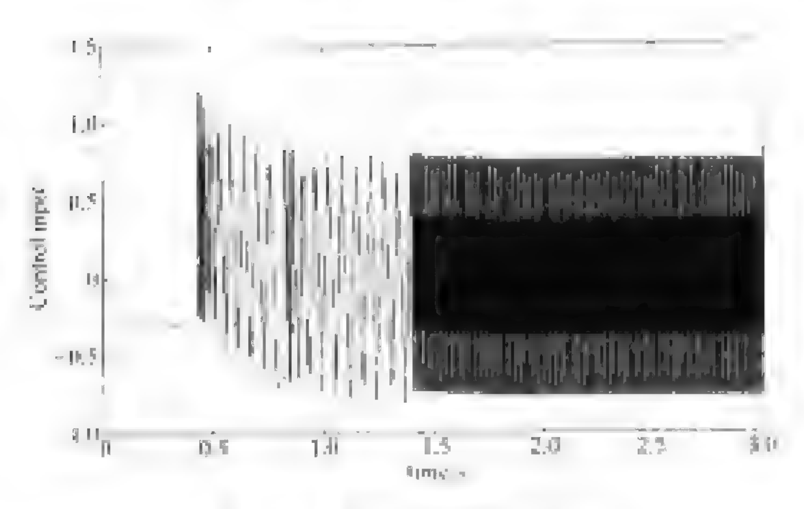


图10-35 控制输入信号(M=1)

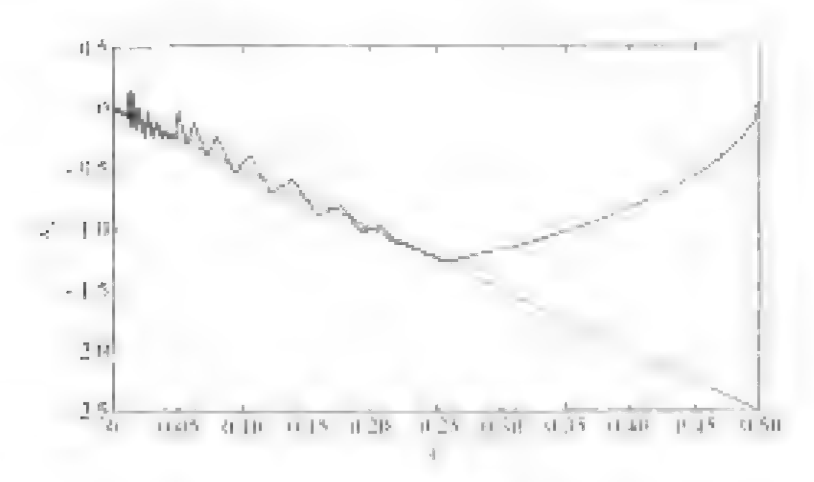


图10-36 相轨迹(M=1)

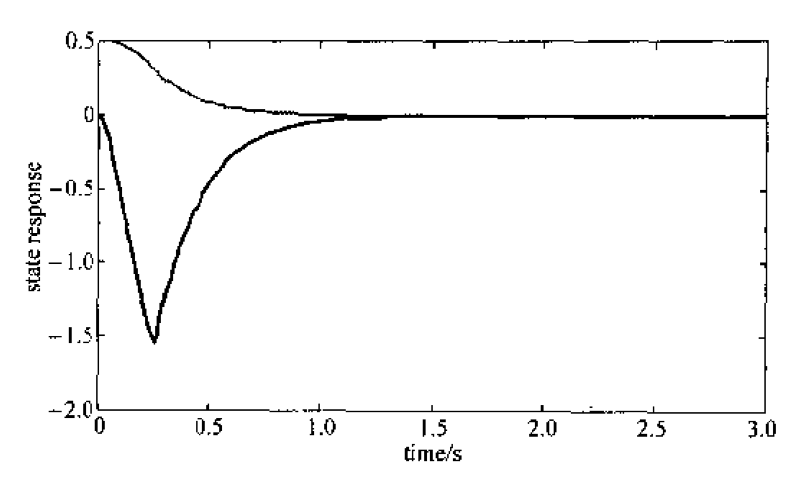


图 10-37　状态响应($M=4$)

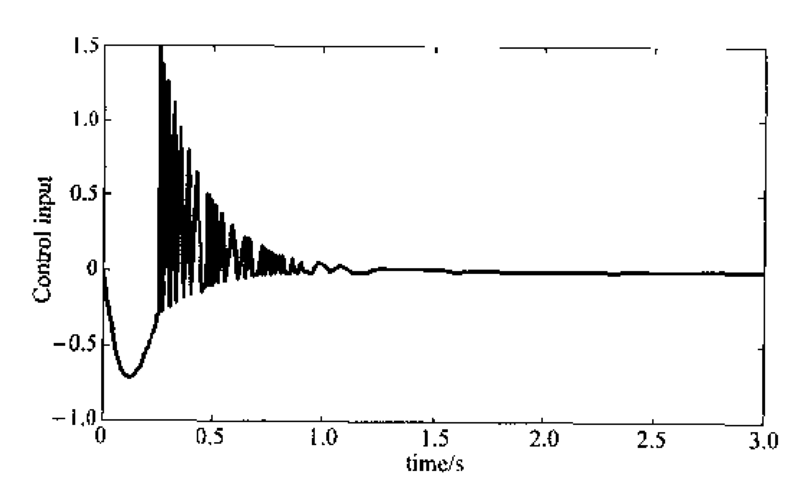


图 10-38　控制输入信号($M=4$)

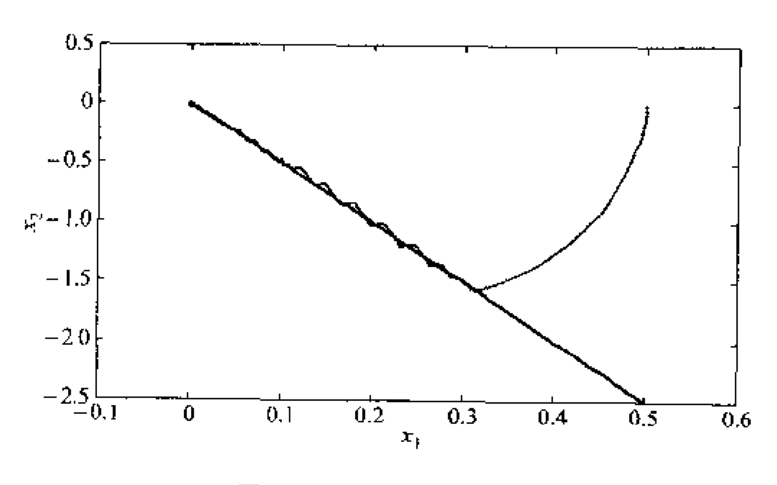


图 10-39　相轨迹($M=4$)

仿真程序：

(1) Simulink 上程序(如图 10-40 所示)：chap10_7sim.mdl

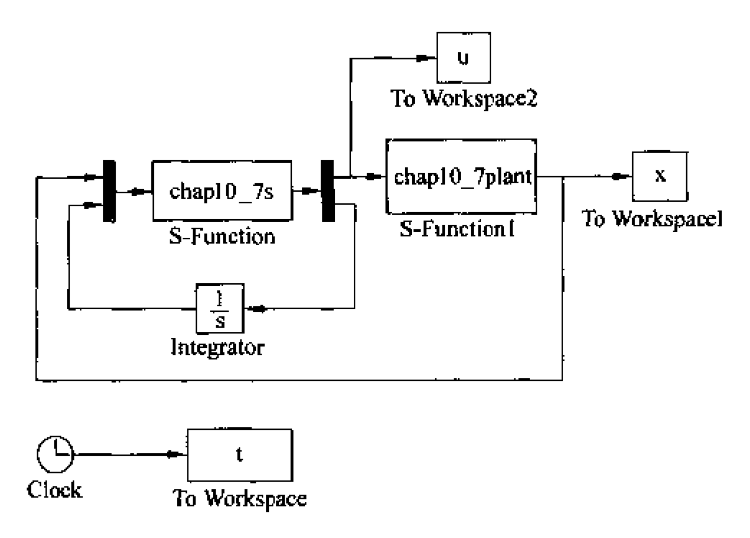


图 10-40　主程序图

(2) 控制器 S 函数程序:chap10_7s.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 0;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 3;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 1;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0 0];

function sys = mdlOutputs(t,x,u)
m = 1.65;
l = 0.28;
D = 0.01;
I = 1/3 * m * l^2;
G = 0.52920;

A = [0 1;0 -D/I];
```

```
B=[0;G/I];

x=[u(1);u(2)];

C=[5 1];
rou=C*x;

M=3;   %Used method
if M==1          %SVSS
   K=abs(inv(C*B)*C*A*x)+1.0;
   ut=-K*sign(rou);
elseif M==2      %EVSS
   Ke=1.0;
   ueq=-inv(C*B)*C*A*x;
   ud=-Ke*sign(rou);
   ut=ueq+ud;
elseif M==3      %IVSS
   Ki=1.0;
   ueq=-inv(C*B)*C*A*x;
   ud=-Ki*abs(u(3))*sign(rou);
   ut=ueq+ud;
elseif M==4      %WIVSS
   Kw=5;
   Kf=-5;
   p=u(3);
   z=Kf*p+rou;
   ueq=-inv(C*B)*C*A*x;
   ud=-Kw*abs(p)*sign(rou);
   ut=ueq+ud;
end

sys(1)=ut;
sys(2)=rou;
if M==4
    sys(2)=z;
end
```

(3) 被控对象 S 函数程序:chap10_7plant.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
```

```
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 2;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 2;
sizes.NumInputs       = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0   = [0.5;0];
str = [];
ts   = [0 0];

function sys = mdlDerivatives(t,x,u)
m = 1.65;
l = 0.28;
D = 0.01;
I = 1/3 * m * l^2;
G = 0.52920;

sys(1) = x(2);
sys(2) = - D/I * x(2) + G/I * u;
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

（4）作图程序：chap10_7plot. m

```
close all;

figure(1);
plot(t,x(:,1),'r');
xlabel('time(s)');ylabel('state response');

figure(2);
c = 5;
plot(x(:,1),x(:,2),'r',x(:,1),- c * x(:,1),'b');
xlabel('x1');ylabel('x2');

figure(3);
plot(t,u,'r');
xlabel('time(s)');ylabel('control input');
```

10.6　模型参考滑模控制

10.6.1　系统描述

被控对象为二阶 SISO 系统：

$$\ddot{y} = a(t)\dot{y} + b(t)u(t) + d(t) \tag{10.85}$$

其中 $b(t)>0$，$d(t)$ 为外部干扰，$|d(t)|<D$。

参考模型为二阶系统：

$$\ddot{y}_m = a_m\dot{y}_m + b_m r(t) \tag{10.86}$$

10.6.2 滑模控制器设计

模型跟踪误差为 $e=y-y_m$，则 $\dot{e}=\dot{y}-\dot{y}_m$。

滑模函数设计为

$$s = \dot{e} + ce \tag{10.87}$$

滑模控制律为

$$u = \frac{1}{b(t)}(-c|\dot{e}|-|b_m r|-D-\eta-|a_m\dot{y}_m|-|a\dot{y}|)\mathrm{sgn}(s) \tag{10.88}$$

其中 $\eta>0$。

稳定性分析

定义 Lyapunov 函数为

$$V = \frac{1}{2}s^2 \tag{10.89}$$

由式(10.87)得

$$\dot{s} = \ddot{e} + c\dot{e} = \ddot{y} - \ddot{y}_m + c\dot{e} = a\dot{y} - a_m\dot{y}_m + bu - b_m r + d + c\dot{e}$$

将滑模控制律式(10.88)代入上式，得

$$\dot{s} = a\dot{y} - a_m\dot{y}_m - (c|\dot{e}|+|b_m r|+D+\eta+|a\dot{y}|+|a_m\dot{y}_m|)\mathrm{sgn}(s) - b_m r + d + c\dot{e}$$

则

$$\begin{aligned} s\dot{s} &= a\dot{y}s - a_m\dot{y}_m s - (c|\dot{e}|+|b_m r|+D+\eta+|a\dot{y}|+|a_m\dot{y}_m|)|s| - b_m rs + ds + c\dot{e}s \\ &= a\dot{y}s - |a\dot{y}||s| - a_m\dot{y}_m s - |a_m\dot{y}_m||s| + c\dot{e}s \\ &\quad - c|\dot{e}||s| - b_m rs - |b_m r||s| + ds - D|s| - \eta|s| \leqslant -\eta|s| \end{aligned}$$

即

$$\dot{V} \leqslant -\eta|s| \tag{10.90}$$

采用饱和函数代替符号函数，可消除抖振。饱和函数设计为

$$\mathrm{sat}(s) = \begin{cases} 1 & s > \delta \\ s/\delta & |s| \leqslant \delta \\ -1 & s < -\delta \end{cases} \tag{10.91}$$

其中 $\delta>0$。

10.6.3 仿真实例

被控对象为

$$\ddot{x} + a\dot{x} = bu(t) + d(t)$$

其中 $a=25$，$b=133$，$d(t)=10\sin(t)$。

参考模型为

$$\ddot{y}_m + a_m \dot{y}_m = b_m r(t)$$

其中 $a_m=20$，$b_m=100$，$r=\sin(\pi t)$。

采用控制律式(10.88)，取 $D=10$，$\eta=0.05$，$c=10$，$M=1$ 时为采用符号函数的控制律，$M=2$ 时为采用饱和函数的控制律，饱和函数中取 $\delta=0.02$，系统初始状态为[1.5 0]，仿真结果如图 10-41～图 10-44 所示。

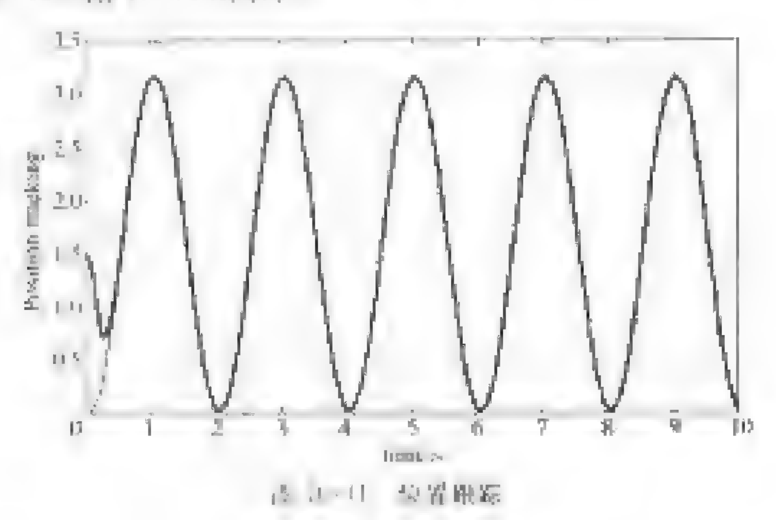


图 10-41 位置跟踪

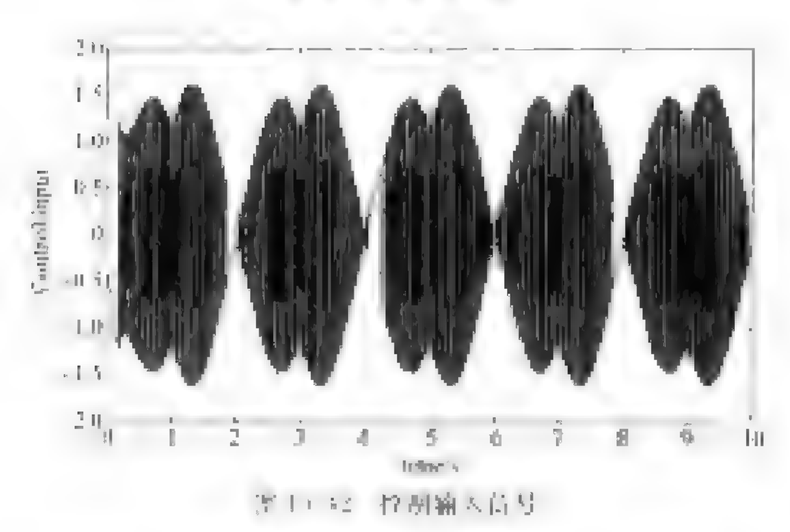


图 10-42 控制输入信号

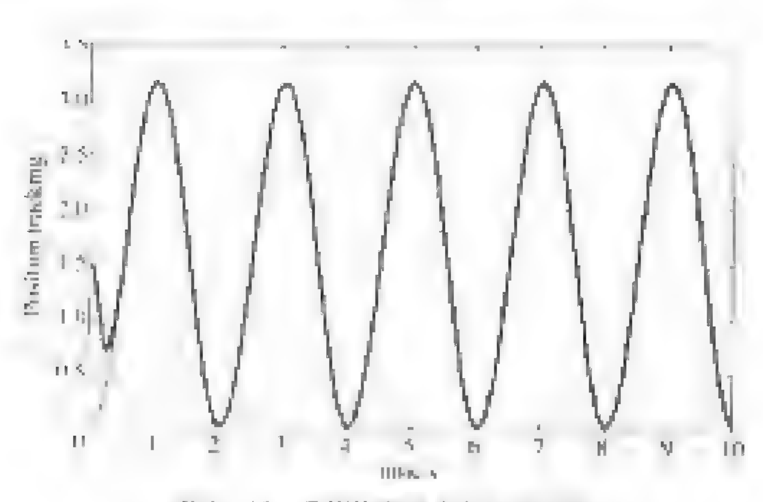


图 10-43 采用饱和函数的位置跟踪

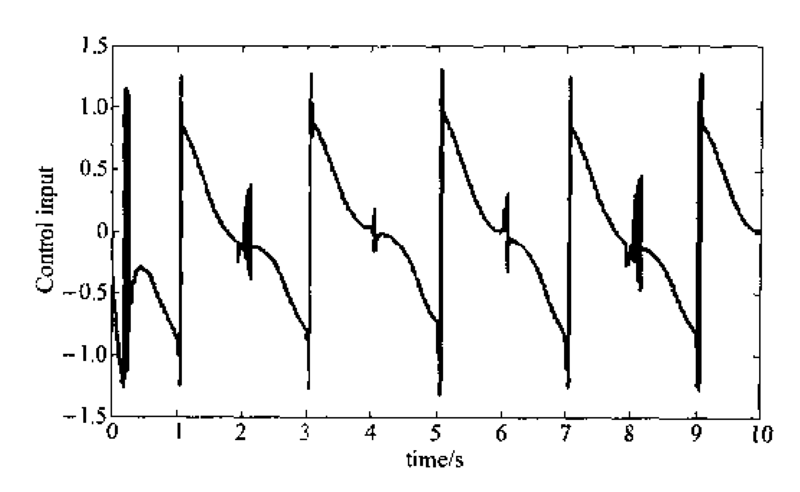


图 10-44 采用饱和函数的控制输入

仿真程序：

(1) Simulink 主程序(如图 10-45 所示)：chap10_8sim. mdl

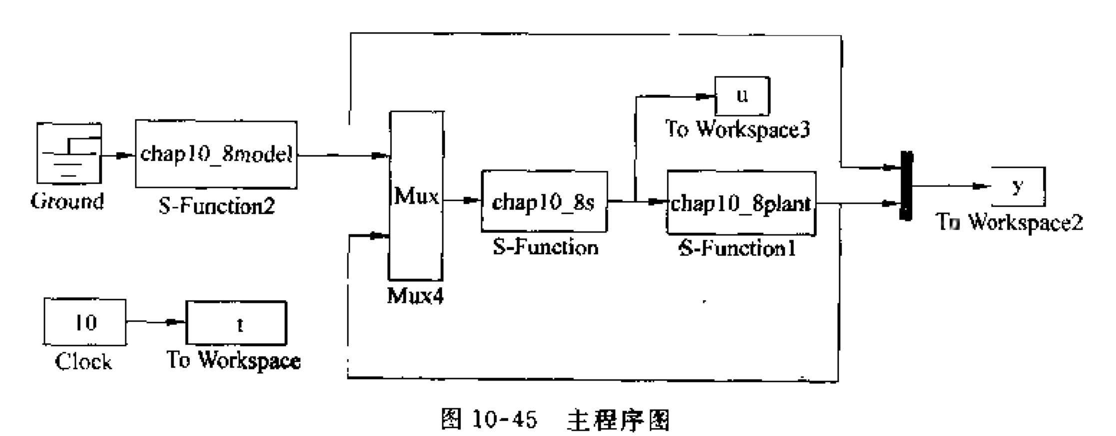


图 10-45 主程序图

(2) 控制器 S 函数程序：chap10_8s. m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
```

```
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 1;
sizes.NumInputs       = 4;
sizes.DirFeedthrough  = 0;
sizes.NumSampleTimes  = 1;
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0 0];

function sys = mdlOutputs(t,x,u)
a = 25;b = 133;
am = 20;bm = 100;
D = 10;
C = [10,1];
ym = u(1);
y = u(3);

e = y - ym;
de = u(4) - u(2);
E = [e;de];
s = C * E;

r = sin(pi * t);
xite = 0.02;

wt = 1/b * (C(1) * abs(de) + abs(bm * r) + abs(am * ym) + abs(a * y) + D + xite);

M = 2;
if M == 1
  ut = - wt * sign(s);
elseif M == 2
    delta = 0.02;
    if s>delta
       sats = 1;
    elseif abs(s)<= delta
       sats = s/delta;
    elseif s< - delta
       sats = - 1;
    end
  ut = - wt * sats;
end
sys(1) = ut;
```

(3) 参考模型 S 函数程序：chap10_8model.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
    sys=mdlDerivatives(t,x,u);
case 3,
    sys=mdlOutputs(t,x,u);
case {2,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [0,0];
str = [];
ts  = [0 0];

function sys=mdlDerivatives(t,x,u)
am=20;
bm=100;
r=sin(pi*t);

sys(1)=x(2);
sys(2)=-20*x(2)+100*r;

function sys=mdlOutputs(t,x,u)
sys(1)=x(1);
sys(2)=x(2);
```

(4) 被控对象 S 函数程序：chap10_8plant.m

```
function [sys,x0,str,ts] = spacemodel(t,x,u,flag)

switch flag,
```

```
case 0,
    [sys,x0,str,ts] = mdlInitializeSizes;
case 1,
    sys = mdlDerivatives(t,x,u);
case 3,
    sys = mdlOutputs(t,x,u);
case {2,4,9}
    sys = [];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end

function [sys,x0,str,ts] = mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates    = 2;
sizes.NumDiscStates    = 0;
sizes.NumOutputs       = 2;
sizes.NumInputs        = 1;
sizes.DirFeedthrough   = 1;
sizes.NumSampleTimes   = 1;
sys = simsizes(sizes);
x0 = [1.5;0];
str = [];
ts = [0 0];

function sys = mdlDerivatives(t,x,u)
a = 25;
b = 133;

sys(1) = x(2);
sys(2) = - a * x(2) + b * u + 10 * sin(t);
function sys = mdlOutputs(t,x,u)
sys(1) = x(1);
sys(2) = x(2);
```

(5) 作图程序：chap10_8plot.m

```
close all;

figure(1);
plot(t,y(:,1),'r',t,y(:,3),'b');
xlabel('time(s)');ylabel('Position tracking');

figure(2);
plot(t,u(:,1),'r');
xlabel('time(s)');ylabel('Control input');
```

参考文献

1. Lu Y S, Chen J S. Design of a global sliding-mode controller for a motor drive with bounded control. International Journal of Control, 1995, 62(5): 1001～1019
2. 申铁龙. 机器人鲁棒控制基础. 北京:清华大学出版社,2000
3. Kang B P, Ju J L. Sliding mode controller with filtered signal for robot manipulators using virtual plant/controller. Mechatronics, 1997, 7(3): 277～286
4. Lin F J, Shyu K K. Novel sliding mode controller for synchronous motor drive. IEEE Transactions on Aerospace and Electronic Systems, 1998, 34(2): 532～542
5. Tahara J I, Tsuboi K, Sawano T, Nagata Y. An adaptive VSS control method with integral type. Switching gain. Proceedings of the IASTED International Conference Robotics and Applications, Florida, USA,2001, 106～111
6. Song J B, Ishida Y. A Robust Sliding Mode Control for Pneumatic Servo Systems. International Journal Engineering Science, 1997, 35(8): 711～723